RTOS PEARL

Integriertes
Echtzeit-
Multitasking-
Programmiersystem

HEISE

Alle Rechte vorbehalten. Dieses Handbuch und die dazugehörigen Programme dürfen weder in Teilen noch als Ganzes in irgendeiner Form (Druck, Fotokopie oder in irgendeinem anderen Verfahren) ohne schriftliche Genehmigung des Verlags Heinz Heise GmbH reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Haftungsausschluß: Der Verlag Heinz Heise GmbH übernimmt keinerlei Haftung oder Garantie bezüglich des Inhalts dieser Veröffentlichung und verneint insbesondere jegliche Gewährleistung der Marktfähigkeit oder der Eignung des Handbuchs oder der dazugehörigen Programme für irgendeinen bestimmten Zweck. Weiterhin behält sich der Verlag Heinz Heise GmbH das Recht vor, diese Veröffentlichung jederzeit ohne jegliche Informationspflicht zu revidieren und darin Veränderungen vorzunehmen.

Hinweis für Anwender: Es wird nicht garantiert, daß dieses Handbuch in allen Einzelheiten auf die dazugehörige Software anzuwenden ist. Falls auf der beiliegenden Diskette eine Datei mit dem Namen LIESMICH enthalten ist, so werden darin bekanntgewordene Abweichungen beschrieben.

Copyright (c) 1987 für RTOS/UH-PEARL:

Prof. Dr.-Ing. W. Gerth
Hannover

Copyright (c) 1987 für Monitor:

Verlag Heinz Heise GmbH
Hannover

Alle Rechte dieser Ausgabe

Verlag Heinz Heise GmbH
Bissendorfer Straße 8
3000 Hannover 61

I N H A L T S V E R Z E I C H N I S

BETRIEBSSYSTEM RTOS-UH

Vorwort	A-0
Schnellkurs	A-I
Datenstationen und Interrupts	A-II
Dynamische Speicherverwaltung	A-III
Interpretation von Fehlermeldungen	A-IV
Einige technische Daten	A-V

BEDIENUNG DES SYSTEMES

Allgemeines über den Bedieninterpreter	B-I
Alphabetisch geordnete Bedienbefehle	B-II
Texteditor: Siehe Bedienbefehl 'ED'	B-II
Einplanungen für Bedienbefehle	B-III

PROGRAMMIEREN IN PEARL

Sprachumfang des UH-PEARL	C-I
Compiler Hilfs- und Test-modes	C-II
Schlüsselwortliste und Einbaufunktionen	C-III
Aufruf von Assembler-/FTN-Unterprogrammen	C-IV
Fehlermeldungen zur Compile-zeit	C-V
Fehlermeldungen zur Laufzeit	C-VI

SYSTEMPROGRAMMIERUNG IN 68000-Maschinencode

Beschreibung des Assemblers	D-I
Fehlermeldungen des Assemblers	D-II
Assemblerprogramme unter RTOS-UH	D-III

INTERNER AUFBAU VON RTOS-UH

Systemtraps, Übersicht, Einzelbeschreibung	E-I
Datenablage auf RTOS-UH-Disketten	E-II
Aufbau des Communication-elementes	E-III
PEARL-Assembler Unterprogrammanschlüsse	E-IV
Ein-/Ausgabe von Assemblerebene aus	E-V
Ergänzung eigener Ein-/Ausgabetreiber	E-VI
Die Scheibenstruktur von RTOS-UH	E-VII

ANHANG

Besonderheiten Ihrer RTOS-Implementierung	F-I
Monitor-Programme	F-II
Liste der Implementierungsscheibe	F-III
Schlagwortindex	

WER MACHT UND MACHTE WAS ?

Am Betriebssystem RTOS-UH wirkten mit:

Prof. Dr.-Ing. W. Gerth, Hannover:

PEARL-Compiler
UH-Assembler
Nukleus
System-Shell
Lader
Editor
Hyperprozessor (Ohne Float-arithm.)
UH-Filemanager, ED-Filemanager
Implementierungsscheiben: ELTEC, GWK, FORCE.

Dipl.-Ing. R. Hausdörfer, Hannover:

Float-arithmetik im Hyperprozessor

Dipl.-Ing. R. Arlt, Hannover:

UH-Filemanagererweiterung 'hierarchisch'
UH-Filemanager-hardwaresektion für ATARI+FORCE WFC 1
Winchesterfilehandler für FORCE WFC 1 und ATARI
FTN-77 Compiler

cand. ing. K. Koerth, Hannover:

Atari-Implementierungsscheibe.
Atari-Graphik-Hilfsfunktionen.

cand. ing. A. Domeyer, Hannover:

Atari-Implementierungsscheibe, Monitoranpassung.

Dipl. Ing. B. Kroll, Hannover

Atari-Basisgraphikfunktionen (Farbe)
Binär READ/WRITE

Dipl. Ing. U. Bartels, Hannover:

MATH-package.

Dipl. Ing. A. Hadler, Hannover:

Implem. Scheibe Commodore Amiga

Real-Time-Operating-System Universität Hannover. Vorwort.
--------*---------*------*----------*----------

RTOS-UH wurde am Institut für Regelungstechnik der Universität Hannover entwickelt. Gemäß der Zielsetzung des Institutes entstand ein Multitasking-Echtzeit-Betriebssystem für Probleme der modernen Prozeßautomatisierung. Dabei wurde ganz bewußt der Schwerpunkt auf Laufzeiteffizienz und Kompaktheit gelegt. Nach meiner Vorstellung sollen damit fertige in Hochsprache (PEARL) codierte Hochleistungs-Automatisierungs-Systeme für ganz unterschiedliche Rechnersysteme realisiert werden - und zwar von Fachleuten, denen die parallele Programmbearbeitung geläufig ist!

Die große Stärke dieses Systemes liegt in der Vielzahl von Computern, auf denen es inzwischen läuft, vom teuren VME-System bis zum preiswerten Personalcomputer, ja sogar auf Single-Eurokarten mit 68008-Prozessor existieren Varianten von RTOS-UH!

Soweit es ging, wurde auch ein Mindestmaß an Komfort berücksichtigt. Es sollte jedoch nie vergessen werden, daß der Komfort eines Interpretersystemes (mit ca. 10-fach langsamerer Ablaufgeschwindigkeit) mit Compilern kaum zu erreichen ist.

Es gibt das Betriebssystem RTOS-UH seit 1980. Seit etwa Herbst 1984 haben wir es an Interessenten für den Preis der Selbstkosten ausgeliefert. Nunmehr (Dez. 1986) dürften etwa 2000 legitimierte Kopien existieren. Einige davon befinden sich in der Hand von Heimcomputernutzern. Inzwischen werden viele schon bemerkt haben, daß sich Basic- oder Pascalprogramme mit etwas Übung direkt in PEARL umsetzen lassen. Damit erkennt man jedoch nur den klassischen Single-Taskbereich, die eigentliche Intention sowie Leistung unseres Systemes erkennt man nur beim Multitasking.

Zum Einstieg haben wir einen Miniaturschnellkurs vorbereitet und dazu gleich eine fiktive Multitaskingaufgabe genommen. Sie finden die Informationen unter A-I.

Mit dem System ist mit kleineren Einschränkungen auch ein Mehrnutzerbetrieb möglich. Man muß sich allerdings dazu klarmachen, daß der Prozeßumschalter von RTOS-UH nur ereignisgesteuert mit festen Prioritäten arbeitet. Es gibt keine Zeitscheiben oder ähnliches, die das System für die Prozeßautomatisierungs-Aufgaben schwerfällig machen. Dafür kann ein 'Dauerrenner' mit hoher Priorität und ohne Wartezustände durchaus andere Nutzer völlig paralysieren. Mit Hilfe der Funktion 'SHARE' können allerdings auch gleichpriorisierte Tasks im Timesharing laufen.

Zum Schluß noch eine dringende Bitte:

Wir sind nur eine sehr kleine Gruppe und können naturgemäß nicht mit den großen Softwarefirmen konkurrieren. Es ist uns daher auch nicht möglich, selbst einen Beratungsdienst bereitzustellen. Sie sollten sich mit Fragen an Ihren Lieferanten wenden, der sich in kniffligen Fällen bei uns melden wird.

Für die Mitteilung (mit Beispiel) echter Systemfehler sind wir in jedem Fall dankbar, damit diese schnellstmöglich abgestellt werden können.

Uni Hannover, Prof. Dr.-Ing. W.Gerth, im Dezember 1986.

This page intentionally left blank.

Einschalten.

Systeme mit Monitorprogramm erwarten nach Meldung des Monitors die Eingabe von 'RTOS', andere starten RTOS-UH ohne Aufforderung.

Es dauert bis zu 20 Sekunden bevor überhaupt etwas auf dem Sichtgerät passiert. In dieser Zeit konfiguriert sich das Betriebssystem (Automatic linking) an Hand der im E-PROM oder Bootbereich zusammengestellten Systemkomponenten. Passiert auch nach längerer Zeit nichts, so muß die Hardware (Terminal auf 9600 Baud?) untersucht werden. Ansonsten erscheint die Kopfzeile mit Konfigurations- und Lizenzhinweisen, abgeschlossen mit der Meldung RESET. Wann immer RESET erscheint, handelt es sich um einen sogenannten 'Kaltstart'. (Das bedeutet später, daß alles zum Teufel ist..)

Erste Aktion.

Bevor sich nun bei CP/M-Freunden Enttäuschung breitmacht, weil bei Anschlag der Tastatur gar nichts passiert, sollte man sich mit der ersten Besonderheit von RTOS-UH vertraut machen: Wir als Nutzer sind nicht die Herrscher im System sondern eher nur ein von Zeit zu Zeit störendes externes Ereignis, auf das es prioritätsgerecht reagieren wird.

Was aber tut er zur Zeit? Der Prozeßumschalter hat festgestellt, daß die einzige lauffähige 'Task' die Leerlaufaktivität #IDLE ist und läßt diese 'arbeiten'. IDLE ist immer lauffähig.

Um ihn von seiner geliebten IDLE-task zu trennen, fabrizieren wir einen Interrupt durch Anschlag von

Ctrl A oder Ctrl B oder Ctrl C oder 'Break' (Undo bei Atari).

Mit Ausnahme von Break/Undo, dort dauert es 2 sek., erscheint sofort nach Anschlag der Prompt '*' und zeigt Eingabebereitschaft an.

----> Wann immer wir Bedienfunktionen des Rechners benötigen, muß eine der obigen Tasten, etwa Control A, angeschlagen werden.

Mit dem Erscheinen des Sternes, übergibt der Dispatcher (Prozeßumschalter) gleich wieder an IDLE. Dennoch können wir jetzt eine Eingabe machen. Wir sind nämlich mit dem Bedieninterpreter (Shell) des Systemes verbunden.

----> Die Sprache des Interpreters darf nicht verwechselt werden mit einer PEARL-Programmierung, auch wenn es viele ganz ähnliche oder gar gleiche Anweisungen gibt.

Die nun zulässigen Eingaben sind in B-I bzw. B-II beschrieben. Wir wählen das Kommando

L (List all Tasks)

und schließen es mit Carriagereturn ab.

Möglich wäre etwa folgende Liste:

```
00001986  +FFF/0  RUN   TWS=00000D36 PC=00081192  #IDLE
000019D0  -001/0  DORM  TWS=00000D92 PC=00000000  #EDFMN
00001A1A  -00A/0  SCHD  TWS=00000E00 PC=00081DEC  #ERROR
00001A64  -002/0  DORM  TWS=00000E5C PC=00000000  #VDATN
00001AAE  -007/0  RUN   TWS=00000EB8 PC=00082C84  #USER0
00001AF8  -006/1  DORM  TWS=000010CE PC=00000000  #USER1
00001B42  -002/0  DORM  TWS=000012E4 PC=00000000  #XCMMD
00001B8C  -001/0  DORM  TWS=000014FA PC=0008C9A0  #UHFX6
00001BD6  -005/0  RUN   TWS=000016B0 PC=0008F4A8  #ACIA1
00001C20  -005/0  DORM  TWS=00001716 PC=00000000  #ACIA2
```

Dabei bezeichnet die erste Adresse den Ort, an dem der sogenannte 'Taskkopf' im verwalteten RAM zu finden ist. In der zweiten Spalte finden wir Priorität/Usernummer des Prozesses. Negative Prioritäten sind für den Hochsprachprogrammierer unerreichbar hoch. Der angezeigte Laufzustand (RUN, DORM, SUSP etc.) ist beim 'L'-Kommando genauer beschrieben.

Groß oder klein?

Im Gegensatz zum PEARL-Compiler akzeptiert der Bedieninterpreter alle Kommandos sowohl in Groß- als auch in Kleinschreibung. Aber man beachte dabei, daß Filenamen immer gleich geschrieben werden müssen! Groß definiert - groß referenziert.

Nun können wir z.B. den Befehl 'S' eingeben (natürlich erst nach erneutem Anschlag von Ctrl A !) und sehen, wie der verwaltete Speicherbereich aufgeteilt ist.

PEARL-Programmentwicklung.

Folgende Aufgabe sei angenommen: Ein Rechenprozeß ('Task') soll fortwährend die Zahlen 1.0, 2.0, 3.0, 4.0, etc. sowie deren Kehrwerte und Quadrate ausgeben. Ein zweiter unabhängiger Rechenprozeß soll jede Sekunde in einem File notieren, welcher Wert gerade bearbeitet wird und den ersten Prozeß nach 100 Sekunden anhalten.

Festlegung eines 'Working-directories' ('WD').

Um für den Editor, Compiler, Lader einfachere Ansprechwege zu erschließen, geben wir den Befehl 'Change directory':

```
CD ED:; (Der Doppelpunkt bedeutet: 'ED' ist eine Datenstation)
        (Nur vorläufig wird auch noch der Punkt akzeptiert)
```

in den Bedieninterpreter. (Nach Ctrl A, Sie wissen schon....) Der Befehl wird vom System mit WD=ED:- bestätigt. Die Vereinbarung bleibt bis zum neuen CD-Befehl oder Kaltstart erhalten.

Einloggen in den Editor.

Wir geben das Kommando ED (nach Ctrl A!) ein und finden uns im Nu im Bildschirmeditor des Systemes wieder. Der Filename wurde vom System 'defaultiert', wir können (s. 'ED'-Kommando) aber auch einen frei gewählten Filenamen wählen, etwa ED TEST.

Erscheint ein chaotisches Bild, so geben Sie gleich die Zeichen ESC X (nacheinander anschlagen, ohne Ctrl A) ein, um den Editor sofort wieder zu verlassen. Wir versuchen eine Umparametrierung mit Hilfe des Bedieninterpreters:

SD A1:+1 01 oder SD A1:+1 02 oder SD A1:+1 03

und loggen erneut mit ED (bzw. ED TEST) in den Editor ein. Hilft auch das nicht, so muß ggf. das Terminal auf den Televideo oder VT-52 kompatiblen Mode umgestellt werden. Standardmäßig ist der Editor auf den Televideo-typ eingestellt. Solange wir im Editor sind, ist übrigens der Zugang zum Bedieninterpreter verwehrt, weil die Zeichen Ctrl A,B,C vom Editor abgefangen werden. Nur Break/Undo ist für den größten Notfall, wenn man sich total verfranst haben sollte, noch aktivierbar. (Oder:Abort-taste der CPU)

Wir löschen die erste Zeile mit ESC R (B-II: ED) und geben ein:

```
MODULE PROBE;            /* Bezeichner sind max. 24 Zeichen lang*/;
SYSTEM;            /* Leitet Definition der Datenstationen ein */;
       Disp: A1<->;            /* Datenstation der Ein-Ausgabe */;
       File: ED.Daten->;         /* File zum Sammeln der Daten */;
PROBLEM;          /* Leitet den Hardware-unabhängigen Teil ein */;
     SPECIFY Disp DATION INOUT ALPHIC CONTROL(ALL);
     SPC File DATION OUT ALPHIC CONTROL(ALL);
     DCL Ausgabenummer FIXED;    /* Zähler für Anzahl Ausdrücke*/;
     DCL (x,Leer) FLOAT;         /* Aktuelles Argument, Leerzyk*/;
     DCL Erstesmal BIT(1);       /* Für Startvorgang           */;
/*-----------------------------------------------------------*/;
Machs:TASK PRIO 50;            /* Ausführende Task            */;
     x=1.0;                    /* Startwert setzen            */;
     REPEAT;                   /* Unbegrenzte Wiederholschleife */;
       PUT x, x*x, 1.0/x TO Disp BY SKIP,(2)F(20),E(20,7);
       x=x+1.0;
     END;         /* Schleifenende */                          ;
END;  /* Ende der Task Machs */                                 ;
/*-----------------------------------------------------------*/;
Steuer:TASK PRIO 49;           /* Überwachende Task           */;
       IF Erstesmal THEN ACTIVATE Machs;
                         Ausgabenummer=1;
                         Erstesmal=NOT Erstesmal;
                     FIN;   /* Ende If-Bereich */;
       PUT Ausgabenummer,x,Leer   TO File BY F(8),(2)F(20),SKIP;
       IF Ausgabenummer EQ 100  THEN TERMINATE Machs;
                                     PREVENT Steuer;
                                FIN;
       Ausgabenummer=Ausgabenummer+1;
END;   /* Ende der Task Steuer */;
/*-----------------------------------------------------------*/;
Rest: TASK PRIO 100  /* Niedrige Prio f.Restkapazitätsmessung */;
      REPEAT Leer=Leer+1.0; END; END;  /* Keine Wartephasen   */;
/*-----------------------------------------------------------*/;
start:TASK PRIO 48;  /* Start-Task   */;
      Erstesmal='1'B; Leer=0.0;
      ALL 1 SEC ACTIVATE Steuer;  ACTIVATE Rest;
END;   /* Ende der Start-Task */
/*-----------------------------------------------------------*/;
MODEND;      Ende des Modules.
```

Schnellkurs

Wir verlassen den Editor mit ESC X (nacheinander!), retten evtl. den File auf Floppy (s. nächste Seite). Dann aktivieren wir die Bedientask (Ctrl A...) und starten den Compiler:

P (Carriagereturn) PEARL kompilieren (Default-file ED:SI)
P TEST (Carriagereturn) Editierter Filename war TEST.

Bei dieser Parametrierung erzeugt der Compiler das Übersetzungsprotokoll auf unserem Terminal. Genaueres über weitere Parameter (Kein Protokoll etc.) finden Sie unter B-II 'PEARL'. Schon nach wenigen Sekunden ist das Programm übersetzt. Falls es Tippfehler gegeben hat - der Compiler zeigt sie uns an- so müssen wir wieder in den Editor einloggen mit ED bzw. ED TEST. Das korrigierte Programm wird dann erneut übersetzt.

Wenn der Compiler nichts anzumeckern hatte, so können wir nun das ganze Modul in den Speicher laden: (Ctrl A .....)

LOAD (Carriagereturn) Objektcode-Filename wurde defaultiert.

Sofort meldet der Lader, daß er fertig ist und wir geben das Bedienkommando: (Ctrl ...)

LU (Carriagereturn) List User-Tasks.

Nun erhalten wir eine Liste, in der unsere 4 Tasks aufgeführt sind. Um das Programm in Gang zu bringen, müssen wir nur noch die Task 'start' aktivieren: (Uff, Ctrl A - schon wieder ..)

start (Carriagereturn) Wo es keinen solchen Bedienbefehl gibt, wird die gleichnamige Task gestartet.

Jetzt rasen Zahlen über den Bildschirm, aber nach 100 Sekunden ist alles wieder ruhig. Wir schauen uns den File 'Daten' mit Hilfe des Editors an: (...)

ed Daten (Carr.return) oder ed ed:Daten falls kein WD vereinbart

Es ist zu erkennen, daß die Task 'Rest' offensichtlich wesentlich mehr (100 bis 300 mal!) Runden drehen konnte als die doch viel höher prioritierte Task 'Machs'. Die Ursache liegt in einem verborgenen Wartezustand von 'Machs': Das Terminal ist zu langsam und nach einer gewissen Anzahl Zeilen ist der Zwischenpuffer im RTOS-UH voll. Dann bremst RTOS-UH die Task 'Machs' mit der Folge, daß der wartezustandfreie Rechenprozeß 'Rest' blendend bedient werden kann. Wenn Sie jetzt staunen, dann haben Sie ein System mit emuliertem Terminal. Bei solchen Systemen kommt 'Rest' nicht von der Stelle, weil die CPU voll fürs Malen beansprucht wird!!

Jetzt können Sie mit dem Programm ein paar Experimente machen, so können wir z.B. die Ausgabe aus 'Machs' entfernen wollen um dann den Versuch zu wiederholen. Dazu müssen wir zunächst unser Modul aus dem System entfernen. (Nebenbei:'Rest' geht's inzwischen noch blendender, sie läuft und läuft..) Das Entfernen dieses Modules erledigen wir mit:

UNLOAD PROBE* (Carr.return) ('*' = 'Inklusive aller Tasks')

Unschön, wenn wir mal den Stern vergessen haben sollten. Dann müssen wir nämlich anschließend jede einzelne Task namentlich mit UNLOAD entfernen, z.B. Unload Machs,Rest,Steuer,start (Carr.rtn)

Wir loggen nun einfach erneut mit Ed bzw. mit Ed TEST in unseren Quellfile ein, ändern diesen, übersetzen ihn neu und laden ihn genau wie vorhin. Wichtig ist, daß wir vor dem Laden wirklich das Modul entfernt haben, ggf. mit dem LU-Befehl überprüfen, ob alle Tasks verschwunden sind.

Bevor wir nun ein zweites Mal 'start' aktivieren, sollten wir den Ausgabefile 'Daten' zurücksetzen, z.B. mit

Rewind Daten (Carr.-return) (Rewind ed:Daten falls ohne WD)

oder durch Elimination des ganzen Files:

rm Daten (Remove) oder erase Daten (gleichwertig)

Sonst würden die neu produzierten Daten hinter die alten gesetzt. Jetzt können wir wieder die Task 'start' aktivieren. Damit ist ein kompletter Programmentwicklungszyklus abgeschlossen.

Retten des Programmes auf Diskette.

Jetzt darf kein Netzausfall passieren, sonst ist unsere ganze Mühe des Programmschreibens für die Katz gewesen. Aus diesem Grund legen wir nun eine Diskette (hier:Laufwerk 0) ein und formatieren sie neu:

FORM D F0:B5DS80 (Carriagereturn) (siehe B-II 'FORM')

Hier 'D' für Double density, 'B5' für das kompakteste 'B'-Format, 'DS' für Double sided und '80' für 80 Tracks. Die Formatierungsinformation wird beim 'FORM' wie ein Filename übergeben.

Wir sind nun ausnahmsweise für eine ganze Weile aus dem System ausgeblendet, weil das Formatieren bei den meisten Systemen eine sehr zeitkritische Angelegenheit ist. Zunächst werden nacheinander beide Seiten beschrieben, dann wird nach defekten Blöcken gesucht (diese werden aus der Verwaltung gestrichen) und das Ergebnis wird auf dem Terminal ausgegeben. Bei mehr als 9 defekten Blöcken allerdings wird die Bearbeitung der Diskette abgebrochen.

Nun können wir mit dem Befehl

DIR F0: (carriagereturn) (':' gibt an: Kein File, sond. Gerät)

nachsehen, wieviel Platz wir haben. Bei ELTEC-Rechnern ist es oft nur die Hälfte, weil der Jumper 'doublesided' vergessen wurde. Jumper bzw. Brücke anbringen und neu formatieren.

Das Retten unseres Programmes erledigen wir mit COPY:

```
COPY ED:SI > F0:xyz      (Defaultierter Filename für User 0).
COPY TEST > F0:xyz       (File-name war TEST).
```

Später können wir den File jederzeit wieder von der Diskette holen, etwa mit

```
Copy f0:xyz > ed:TEST    (Falls kein Working directory vereinbart)
Copy f0:xyz > A1:        (Ausgabe auf dem Sichtgerät User 0)
```

Files geschlossen??

Zur Einsparung von Diskettenoperationen wird, solange es geht, das Inhaltsverzeichnis und der aktuelle Datenfile teilweise im Speicher gehalten. Die Daten sind daher nur dann auf der Diskette gesichert, wenn alle Files vor dem Herausnehmen geschlossen sind. Geben Sie dem System eine Chance, prüfen Sie, ob Sie die Diskette herausnehmen oder den Rechner abschalten dürfen:

```
CF F0:     (Change Floppy, Meckert, falls ein File noch offen ist)
FILES F0:  Listet alle offenen Files auf, anschließend wird mit
RETURN f0:xyz,f0:abcd ...... jeder einzelne File geschlossen oder

SYNC F0:; CF F0:FORGET   eingegeben. Nun erst Diskette entnehmen.
```

Der Lader, Compiler, Copy etc. hinterlassen normalerweise keine offenen Files. Sie treten darum nur bei Unregelmäßigkeiten und abgebrochenen Programmen auf.

Alles zur gleichen Zeit!

Wenn Sie etwas experimentiert haben, so werden Sie schon bemerkt haben, daß unser System anscheinend tausend Dinge gleichzeitig erledigen kann. Sie können z.B. einen Copy auf die Schnittstelle A2: oder PP: (Centronics) in Gang setzen, und während die Ausgabe läuft, ungehindert das nächste Programm entwickeln. Bei anderen Systemen so gerühmte 'Spooler' sind bei dieser Struktur eigentlich überflüssig. Man muß nur immer darauf achten, daß stets nur eine Operation pro File stattfindet, also nicht den gleichen File sowohl im Editor haben als auch gleichzeitig kompilieren oder kopieren!
Dagegen können Sie ruhig während der Kompilation eines Files den Kompilationslauf eines anderen Files in Gang setzen oder diesen anderen editieren und vielleicht zwanzig andere assemblieren ....

Noch etwas über den Bedieninterpreter.

Bisher haben wir immer nur eine Anweisung in jeder Zeile benutzt. Wir können durchaus mehrere Anweisungen in eine Zeile schreiben und dabei festlegen, ob diese 'gleichzeitig' oder nacheinander zu erledigen sind:

```
P TEST -- load -- start
```

bedeutet: Kompiliere, wenn fertig (und 0 Fehler!) lade, wenn mit laden erfolgreich fertig starte 'start'. Der Sinn der zeitlichen Kettung besteht darin, daß man vielleicht gerne mal irgendwohin

möchte und das Programm in Aktion sehen will, wenn man zurück kommt.....

Dagegen werden bei dem Befehl

```
P TEST;COPY mist > A2:;COPY kaese > PP:
```

gleichzeitig ein Compiler und zwei Kopiervorgänge laufen. Ohne 'SHARE' hat der Compiler höhere Priorität, die beiden 'COPY' laufen nur mit den CPU-Restkapazitäten. Diese können (siehe unser Testprogramm) bei einer Protokollausgabe des Compilers allerdings so groß sein, daß beide Copy mit voller Geschwindigkeit voran kommen können. Nebenbei: A2: ist die Stationsbezeichnung für die zweite serielle Schnittstelle, PP: die Stationsbezeichnung für ein evtl. vorhandenes Parallelport (Centronics). (s. Seite 10).

Fernsteuerung.

Alle Befehle des Bedieninterpreters können auch von PEARL-Programmen aus ausgeführt werden! Sie brauchen dazu nur den Eingabetext mit 'PUT' in die Systemdatenstation 'XC' (Remote-Control) einzuschreiben. Es ist immer eine nette Geste, wenn man im Rechner für den nächsten Nutzer eine Formatieranweisung zur Exekution um 16.00 Uhr o.ä. zurückläßt .....

```
SYSTEM; Bedien:XC;
PROBLEM;      SPC Bedien DATION OUT ALPHIC CONTROL(ALL);
Batman:TASK; PUT 'FORM D F0:B5DS80' TO Bedien BY A,SKIP; END;
MODEND;
```

Nach dem Übersetzen und Laden geben Sie den Bedienbefehl

```
AT 16.00 ACTIVATE Batman
```

und gehen fröhlich pfeifend nach hause ...
Ich hoffe nun doch sehr, daß Sie das nicht wirklich mal machen!
Sinnvoll ist aber die Möglichkeit, sich Blockkommandos schaffen zu können, etwa ein Kommando 'Neu' als PEARL-Programm:

```
Neu:TASK;PUT 'Unload PROBE*;ED--P--Load--start' TO Bedien;END;
```

Nach dem Laden dieser Task braucht man nur noch jeweils den Befehl 'Neu' einzugeben (auch während Machs läuft!) und findet sich im Editor wieder. Sobald man diesen verläßt, wird kompiliert, geladen und gestartet. Geht's denn noch einfacher??

Weitere Nutzer.

An anderen Terminals, die ebenfalls einen Bedieninterfaceanschluß besitzen, ist die Bedienung nicht anders. Allerdings sollte man bei mehr als einem Nutzer sich hierarchisch in das ED-Filesystem einloggen. Sonst riskieren Sie zufällige Namensgleichheit bei den Files.

```
CD ED:Mueller
```

Mit z.B. ED TEST; wird nun mit voller Pathlist ED:Mueller/TEST adressiert. Wenn der andere Nutzer nicht auch Mueller heißt und sich unter anderem Namen einführt, ist sein File 'TEST' einer im anderen Zweig des Baumes.

Das Konsolenterminal (User Ø) hat allerdings vereinzelt doch einen Sonderstatus, z.B. bei Fehlermeldungen in Interruptprozessen.

Haben Sie eine Winchester?

Dann sollten Sie Winch-files immer nur hierarchisch organisieren, sonst steht hinterher alles in der root-Ebene und Sie finden sich garantiert nicht mehr zurecht. Ein Directory müssen Sie jetzt aber explizit einrichten mit z.B.

mkdir HØ:usr/mueller (Das Directory 'usr' existierte schon)

Fortan können Sie etwa nach HØ:usr/mueller/TEST schreiben oder von dort lesen. Greifen Sie häufiger darauf zu, so kann natürlich mit

CD HØ:usr/mueller

der Zugriff vereinfacht werden. Nur müssen Sie dann beim Editieren die ED-files über die volle pathlist ansprechen:

Copy TEST > ED:ge/TEST--ed ed:ge/TEST--copy ed:ge/TEST > TEST

Bei diesem Befehl wird der File von der Winch. geholt, editiert und anschließend wieder zurückgeschrieben. Nicht mehr hübsch?

Rein und Raus?

Auf den nächsten Seiten des Schnellkurses finden Sie noch einige Tabellen zu den Schnittstellen. Die Schnittstellen selbst sind einzeln unter A-II beschrieben.

Unsere 'Teufelchen'.

In allen komplexen Computersystemen lauern kleine Teufelchen und sind darauf aus, Ihnen Fallen zu stellen. Ganz am Ende dieses Schnellkurses haben wir deren übelste RTOS-UH-Tricks aufgelistet.

Interessiert?

Lesen Sie sich doch alle Befehle des Bedieninterpreters einmal durch (B-II). Sie finden dort die detaillierte Information, die hier keinen Platz fand. Einige Bedienbefehle haben Sie ja schon benutzt, wenn auch z.T. nur mit einem Teil der möglichen Parameter. Zu PEARL selbst? Siehe gesondertes Literaturverzeichnis.

Unbedingt lesen sollten Sie die Abschnitte C-I bis C-VI, es betrifft die Möglichkeiten des eigentlichen PEARL-Teiles.

Der Erzeuger von RTOS-UH wünscht Ihnen viel Glück und Erfolg!

Schnittstellen ('DATION') im UH-PEARL Datenstationen

Grundsätzlich existieren zwei Typen von Datenstationen, je nachdem ob sie annähernd mit Prozessorgeschwindigkeit die Daten übertragen können oder nicht.

* D/A-Wandler, Digitalkoppler etc. sind quasi jederzeit bereit. Sie werden darum nicht vom Betriebssystem als Betriebsmittel verwaltet, der Compilercode greift direkt darauf zu.

* Terminalschnittstellen, Druckerports, Floppydiscs etc. sind langsamer als der Prozessor und werden daher nur unter Kontrolle von RTOS-UH zugänglich gemacht. Die Bearbeitung erfolgt ähnlich wie am Postschalter: Es werden 'Warteschlangen' aufgebaut und nur das jeweils vorne befindliche Datenpaket wird zu einem Zeitpunkt bearbeitet.

Die folgende Beschreibung gilt demnach nur für die Stationen des zweiten Types mit Warteschlangen.

Zu jeder Station existiert eine Warteschlangennummer, 'LDN' genannt ('Logical Dation-Number'). Benutzen mehrere anscheinend eigenständige Geräte (z.B. Floppy 0, Floppy 1) gemeinsame Bausteine (Floppykoppler) so werden sie dennoch nur durch eine einzige LDN repräsentiert. Die einzelnen Geräte werden dann durch die sog. Untergliederungsnummer ('DRIVE') unterschieden. Oft wird die Untergliederungsnummer auch nur benutzt, um verschiedene Betriebsarten eines einzigen Gerätes anzuwählen. Die nächst feinere Unterteilung erfolgt durch den 'FILE-namen' oder noch genauer durch eine baumförmige 'path-list'.

Jede Warteschlange besitzt einen 'Bediener', quasi der Beamte am Postschalter. Wir nennen diesen Bediener 'Betreuungstask' der Warteschlange. Die Aufgabe der Betreuungstask besteht einfach nur darin, die Warteschlange möglichst schnell abzubauen. Wie im täglichen Leben so gibt es auch hier 'ganz eilige Kunden', die sich frech nach vorne drängen, evtl. direkt bis zum Schalter. Wie weit man sich vordrängen kann, hängt von der Wichtigkeit des Auftraggebers ab: der Priorität der lese-/schreibwilligen Task. (Der Name der Betreuungstask ist für den Anwender uninteressant, er ist meist analog zur Gerätebezeichnung gewählt.)

In PEARL-Programmen kann man nun den einzelnen Geräten frei wählbare Namen zuordnen. Das geschieht mit Hilfe des 'Systemteiles'. So bedeutet etwa

```
XYZ:LD/5,3/.abcd/efg - ;
```

Die Station heißt 'XYZ', besitzt die Warteschlangennummer '5' und benutzt das Laufwerk no. '3'. Über das 'directory' namens 'abcd' wird der File 'efg' adressiert. Man beachte, daß die pathlist im UH-PEARL durch 'OPEN BY IDF..' veränderbar ist. Durch weitere Zusätze kann auch der Betriebsmode modifiziert werden.

Stationen mit Warteschlangen werden bei PUT und GET benutzt.

Für viele häufig benutzte Stationen gibt es im Bediensystem und im Compiler noch Fertignamen, zu denen LDN und DRIVE automatisch generiert werden, z.B. 'A1' für das Konsolenterminal = LD/0,0/.

- Datenstationen -

Tabelle der DRIVES (nach unten) und LDN (nach rechts) aus Mnemos

Syst.	DR.	0	1	2	3	4	5	6	7	8	9	10
ATARI	0	A1	ED	A2	F0	A3			VO	VI	XC	PP
	1				F1							
	2	B1		B2	H0	B3						
	3			UL								
	4											
	5											
	6	C1		C2		C3						
ELTEC	0	A1	ED	A2	F0		(F0)		VO	VI	XC	
	1				F1		(F1)					
div.	2	B1		B2	F2		(H0)					
	3			UL	F3		(H1)					
	4						(H2)					
	5						(H3					
	6	C1		C2								
FORCE	0	A1	ED	A2	F0	MT			VO	VI	XC	
FPK-2	1				F1							
	2	B1		B2	F2							
	3			UL	F3							
	4											
	6	C1		C2								
FORCE	0	A1	ED	A2	F0	A3			VO	VI	XC	
VME	1				F1							
	2	B1		B2	H0	B3						
+	3			UL	H1							
Motor.	4				H2							
..133	5				H3							
	6	C1		C2		C3						
GWK	0	A1	ED	A2	F0	MT	H0		VO	VI	XC	PP
c't-	1				F1							
68000	2	B1		B2	F2							
	3			UL	F3							
+	4											
VME	5											
	6	C1		C2								
PEARL-	0	A1	ED	A2	F0	MT			VO	VI	XC	PP
Comp.	1				F1							
	2	B1		B2	F2							
	3				F3							
	4											
	6	C1		C2								

Die geklammerte Belegung bei der LDN 5 und den ELTEC-Systemen gilt für den Fall der Bestückung mit dem Winch./Floppy-Interface WFC-1 von FORCE.
Nicht aufgeführte Systeme gemäß Äquivalenzangaben. Tabelle enthält natürlich nicht die evtl. nachkonfigurierten Dations.

Hier erfahren Sie etwas über die hinterhältigsten Tricks der üblen Gesellen - soweit sie uns bekannt sind.

Das Ctrl S - Teufelchen. Dieser Kamerad wartet nur darauf, daß Sie aus Versehen statt Ctrl A einmal das Zeichen Ctrl S anschlagen. Dann kann er nämlich das Terminal totlegen. Das Ctrl S ist an sich sehr notwendig, um etwa rasende Ausgaben anhalten zu können (X-off) - und dafür benutzen wir es ja auch absichtlich. Geben Sie dem Kerl eines auf die Nase und bieten Sie ihm Ctrl Q an (X-on), mehrmals schadet nichts.

Das 'No-scroll'-Teufelchen. Es siedelt nur in einigen besonderen Terminals, die eine 'No scroll'-Taste besitzen. Oft hilft dann Ctrl Q nicht, sondern nur die terminalseitige Aufhebung der No-scroll-Bedingung.

Das 'Input-queue'-Teufelchen. Ein schwer zu bekämpfender Parasit, der den 'Beamten' am Postschalter befällt, er kann den aktuellen Auftrag nicht ausführen. Eine Task veranlaßt zum Beispiel eine Leseoperation ('GET') vom Terminal. Solange auf dem Gerät die erwartete Zahl von Zeichen nicht eingegeben wird, geht's in der Schlange nicht voran. Trotz der hohen Priorität kommt auch die Bedientask nicht durch. Scheinbar reagiert das System auf Ctrl A nicht. Uns bleibt nichts anderes zu tun als Ctrl A anzuschlagen, evtl. die Daten einzugeben oder ersatzweise sooft die Carriagereturn-Taste zu betätigen bis der Eingabeprompt der Shell erscheint.

Der Index-error-Teufel. Ja, dies ist kein ...chen, sondern die Inkarnation der Teufelei schlechthin! Er kann wohl alles Mögliche an Schäden im System anrichten, von zerstörten Zeigern bis zu Systemabstürzen, die erst Stunden später auftreten. Gegen ihn gibt es nur eine Waffe: Die 'T'-Option des PEARL- oder F77-Compilers. Damit wird der unerlaubte Zugriff durch überschrittene Indexgrenzen in fast allen Fällen verhindert. Hat der Bold aber bereits zugeschlagen, sollten Sie von Ihren Files retten, was zu retten ist und die RESET-Taste aktivieren.

Das Bus-error-Teufelchen. Wenn Sie mit 'DM' (s.B-II) einen nicht bestückten Speicherbereich ausgeben wollen, so erfolgt die Meldung 'BUS ERROR' und die Bedientask stellt anscheinend ihre Tätigkeit für Sie ein, denn nach jedem Ctrl A gibt's gleich wieder einen Fehlerabbruch. Nun könnte man sooft Ctrl A anschlagen wie insgesamt nicht erreichbare Bytes ausgegeben werden sollen. Hier hilft es aber viel besser, wenn Sie die BREAK/Undo-Taste betätigen.

Das Doppellade-Teufelchen. Seine Spezialität besteht darin, uns zu einem unüberlegten LOAD-Befehl zu verführen, obwohl eine gleichnamige Task noch in der Verwaltung existiert. Mit jeder Aktivierung der Task erwischen wir dann immer nur die erste (meist alte) und wundern uns, daß Programmänderungen ohne Wirkung geblieben sind. Wir sollten darum stets sicher gehen, daß Doppelladen nicht auftreten kann.

This page intentionally left blank.

Bei diesen Stationen handelt es sich entweder um emulierte Terminals oder um serielle Schnittstellen (RS 232c, RS 458 etc.). Für Ay,By,Cy gilt - bei gleichem y - stets eine gemeinsame LDN, die Buchstaben A,B,C werden also lediglich in eine Untergliederungsnummer (=Betriebsart) umgewandelt. Die Datenstation UL kennzeichnet eine Sonderbetriebsart der Station mit y=2.

Da für beide Übertragungsrichtungen nur eine Warteschlange zur Verfügung steht, ist ein echter Vollduplexbetrieb mit diesen Stationen nicht möglich. Solange also etwa eine Eingabe 'hängt', tritt in der Warteschlange Stillstand ein. Um wenigstens die Operation 'mal sehen, ob was da ist' für die Eingabe machen zu können wurde die Betriebsart C geschaffen.

A) AUSGABE.

Es gibt zwischen Ax,Bx,Cx und UL keinen funktionellen Unterschied. Die Ausgabe kann vom Empfänger jederzeit mit Hilfe eines ausgesendeten 'X-off' (Ctrl_S) angehalten werden. Mit Hilfe des 'X-on' (Ctrl_Q) wird dann die Wiederaufnahme der Sendung angefordert. (X-on/X-off-Protokoll). Auch emulierte Terminals, so etwa beim c't 68000 oder Atari, können auf diese Weise angehalten werden.

In manchen Systemen (ELTEC, ELZET-80) kann über besondere Bedienbefehle die Baudrate verändert werden.

Ebenso wird in einigen Systemen zusätzlich zum Softwareprotokoll ein Hardwareprotoll (DTR) unterstützt.

Wichtig: Die zugehörige I/O-Task 'merkt sich', ob sie als letzte Anforderung über A oder über B,C angesprochen wurde. Dies steuert ihr Verhalten hinsichtlich des unten beschriebenen Eingabekanales.

B) EINGABE ohne Initiative der I/O-Task

1) Die Station war zuletzt als Ax-Station in Benutzung.

   Die unaufgefordert empfangenen Daten werden in einen implementierungsabhängigen Eingabepuffer (Ringpuffer) geschrieben. Es wird keine Reaktion zum Sender ausgesendet.

2) Die Station war zuletzt als Bx oder Cx in Benutzung.

   Auch hier werden die unaufgefordert empfangenen Daten in den begrenzten Ringpuffer genommen, allerdings wird nun für jedes Zeichen, das auf den Puffer geht, ein X-off zum Sender zurückgesendet, um diesen zu stoppen.

C) EINGABE mit Initiative der I/O-Task.

1) Expliziter Lesebefehl für die Station Ax.

Die evtl. im Eingaberingpuffer befindlichen Daten werden eliminiert (Puffer wird gelöscht) und sind damit verloren. Die I/O-Task hängt sich jetzt auf ('SUSP') bis die angeforderte Anzahl von Daten eingetroffen ist. Beim Lesen mehrerer Sätze und ständig laufendem Eingabestrom gehen bei dieser Betriebsart in den 'DORM' - phasen der I/O-Task zwangsläufig Daten verloren, denn es wird kein X-off-Protokoll erzeugt. Die Ax sind als Eingabestationen praktisch nur für die Terminaleingabe des Bedieninterface geeignet, weil es dort gerade erwünscht ist, daß evtl. in der inaktiven Phase angeschlagene Zeichen (Affe, der auf die Tasten hämmerte..) keine Auswirkungen haben.

2) Expliziter Lesebefehl für die Station Bx, UL.

Die im Eingaberingpuffer befindlichen Zeichen werden übertragen. Falls diese bereits die Anforderung erfüllen konnten, so ist die Operation - ggf. mit aufgehobenem Rest im Eingaberingpuffer - beendet.

Beim Lesen von der Station UL, entsprechend der Station B2, werden empfangene Line-feeds nicht übernommen. Diese Option ist hauptsächlich zum Laden von S-Records von einem Host-Rechner geeignet (UL = Up-Load), kann aber auch immer dann eingesetzt werden, wenn vom Host zusätzliche Line-feeds generiert werden ('Aufblähen' des CR zur Kombination CR/LF). Die Station UL ist nur über das Bedieninterface erreichbar, von PEARL her muß sie über LDN/2,3/ angesprochen werden.

Reichen die im Puffer befindlichen Daten nicht aus, so geht die I/O-Task in den Zustand 'SUSP' bis zu dem Zeitpunkt, zu dem die erforderliche Anzahl Daten erreicht wird.

Beim Lesen mehrerer Sätze und einem fortlaufenden Eingabestrom kommt es hier nicht zu einem Datenverlust, da die Betriebsart 'B' das X-on/X-off-Protokoll benutzt und damit auch in den 'DORM'-Phasen der I/O-Task keine Daten verloren gehen. (sofern der Sender rechtzeitig auf das X-off hört!!)

3) Expliziter Lesebefehl für die Station Cx.

Wenn keine Daten im Eingaberingpuffer stehen, so wird dem Auftraggeber nur das Zeichen NUL ($00) mit der Satzlänge 1 (RECLEN = 1) übermittelt und der Auftrag wird als erledigt behandelt.

Stehen im Eingaberingpuffer weniger Zeichen als angefordert, so werden diese übertragen und die erreichte Satzlänge wird dem Auftraggeber mitgeteilt. Der Auftrag gilt danach als erledigt.

Stehen im Eingaberingpuffer ausreichend Zeichen, so wird - unter Erhalt eines evtl. Restes - der Auftrag daraus erledigt.

Auch bei dieser Betriebsart erfolgt die Sendersteuerung mit dem X-off/X-on-Protokoll wie bei der Betriebsart 'B'.

Die Betriebsart 'C' ist also gedacht, um ohne Risiko des Aufhängens das Eingabeport abzufragen, ob denn 'Daten da sind'. Beim Lesen aus Hochsprachprogrammen über 'C' muß der Programmierer selbst auf die NUL abprüfen - dann waren noch keine Daten eingetroffen.

Das Bedienkommando 'COPY' (s. B-II) ist auf das NUL-Zeichen und die Satzlänge 1 abgerichtet: Nach Einlesen des NUL-Zeichens legt sich 'COPY' für 8 millisec. 'aufs Ohr' um danach die Abfrage zu wiederholen.

Mit COPY.W C1:>B2:; COPY.R C2:>B1:;

kann man seinen Rechner in so etwas wie ein Terminal für den angeschlossenen Host verwandeln. Mit Ctrl A wird dann bei Bedarf die Verbindung zum RTOS-UH wiederhergestellt.

Auch das Lesen vom Cx kann zum Aufhängen führen, nämlich dann, wenn die Schlange durch eine liegengebliebene Ausgabe oder durch eine Eingabe der Betriebsart A bzw. B 'verstopft' ist.

BEDIENINTERFACE: Stationsname ist Ax, Bx, Cx, z.B. A1:, C2:, B2:
Ein eventuell angegebener Pfadname wird voll in die Verwaltung übernommen ('S'-Befehl), hat aber keinerlei Funktion.

Bsp: O B2:; DM 1000 2000;
P > B2:; COPY B3:>B1:;

PEARL-PROGRAMM: Systemname ist Ax, Bx, Cx, z.B. A1; C2; B2; Eine eventuell nachgestellte pathlist wird voll in die Verwaltung übernommen ('S'-Befehl), hat aber keine funktionelle Bedeutung.

Bsp:

```
SYSTEM; Output:B3->;
        Inputdevice:B2<-;
        A1;   /* Kurzform username=systemname*/;
......
PROBLEM;
  ......
  SPC A1 DATION INOUT ALPHIC CONTROL(ALL);
  SPC Inputdevice DATION IN ALPHIC CONTROL(ALL);
  SPC Output DATION OUT ALPHIC CONTROL(ALL);
  ......
  GET .... FROM Inputdevice BY ....
  PUT .... TO Output BY ........
```

Die Datenstation BU läßt den direkten Zugriff auf Prozeßperipherie über allgemeine Peripherieadressen oder über den Unterbus (P-Bus, nicht bei allen Systemen implementiert) zu. Zu dieser Station existiert keine Betreuungstask, da über die Peripherieadressen bzw. den Unterbus nur ungepuffert und unquittiert aus-/eingegeben werden kann. Das Betriebssystem ist an dem Transport nicht beteiligt. Die BU-Stationen können also nicht vom Bedieninterface angesprochen werden. In Systemen, die Mehrrechnerbetrieb zulassen (z.B. VME-Bus), können Synchronissationsmaßnahmen unterstützt werden, die wie globale Semaphore wirken.

Benutzung im PEARL-Programm:

Der Systemname ist BU(hexadr8,Zugriffscode), alternativ (bei 68000-er Rechnern, die nur 6-Byte-Adressen verarbeiten können) BU(hex8), wobei der Zugriffscode zweistellig vor der Periperieadresse angegeben werden muß. Die Datenstation gilt als vom Typ 'BASIC'. Die Übertragungsrichtung muß angegeben werden. Das Bitmuster hexadr8 muß in Kodierung und Adresse der aktuell angeschlossenen Hardware entsprechen.

Um möglichst viele unterschiedliche Peripheriekarten verwenden zu können, wurden folgende Zugriffsmöglichkeiten geschaffen:

In der Anweisung

VENTIL: BU(y,x) ->;

bedeuten: y = Adresse der Peripheriekarte (des Peripheriebausteins) in hex8.
x = Kodierung der Zugriffsart, einstellig dezimal

Folgende Kodierungen sind zugelassen:

CODE	Übertragung	wird abgelegt auf
0	P-Bus, WORD oder LSB	
1	8 bit, MSB (MOVE.B)	y
2	16 bit, MSB (MOVE.W)	y, y+1
3	32 bit, MSB (MOVE.L)	y, y+1, y+2, y+3
4	16 bit, MSB (MOVEP.W)	y, y+2
5	32 bit, MSB (MOVEP.L)	y, y+2, y+4, y+6
6	8 bit, LSB (MOVE.B)	y
7	P-Bus, WORD oder MSB	
8	GLOBAL SEMA (TAS)	

Alle einfachen Datentypen sind zugelassen (Ausnahme: Kodierung $08). Bei Float-Variablen wird normalisiert bzw. denormalisiert. CHAR-Variablen werden linksbündig beschrieben und ggf. mit BLANKs aufgefüllt.

Bei Kodierung $08 ist eine BIT(1)-Variable zu verwenden. Die TAKE-Anweisung benutzt den Assembler-Befehl TAS und liefert den Wert '1'B1, wenn der Pseudo-Request erfolglos war (dann stand bereits eine '1' an der getesteten Speicherstelle) oder '0'B1,

wenn der Pseudo-Request erfolgreich war (dann stand bisher eine '0' an der Speicherstelle, die vom TAS-Befehl mit einer '1' überschrieben worden ist).
Die Kodierungen $04 und $05 sind für einige Bausteine aus 8-bit-Prozessor-Familien erforderlich.

BEISPIELE: I) Zugriff über Peripherieadressen

```
SYSTEM;
FUEHLR: BU(0240FFFC) <-;      /* MC 68000:
                  Adr=$40FFFC, Zugriffscode=2 */;
.........
PROBLEM;
SPC FUEHLR DATION IN BASIC;
.........
R: TASK;
   DCL TEMP BIT(16);
   TAKE TEMP FROM FUEHLR;
```

II) Zugriff über den Unterbus

```
SYSTEM;
VENTIL: BU(42) ->;
.........
PROBLEM;
SPC VENTIL DATION OUT BASIC;
.........
T: TASK;
   SEND '008F'B4 TO VENTIL;
```

III) Globale Pseudo-Semaphore

```
SYSTEM;
S1: BU(FF0F0006,8) <->;       /* MC 68020,
                  Adr=$FF0F0006, Zugriffscode=8 */;
.........
PROBLEM;
SPC S1 DATION INOUT BASIC;
.........
INIT: TASK;
   SEND '01'B1 TO S1;
.........
X: TASK;
   DCL FLAG BIT(1);
   .........
   FOR I TO MAX WHILE FLAG REPEAT;
      TAKE FLAG FROM S1;
   END;
   .........                    (geschützter Bereich)
   SEND '0'B1 TO S1;                 (Peudo-Release)
```

VORSICHT
Da sich die Peripherieadressen nicht von Speicheradressen unterscheiden, können durch falsche Adressangaben in der BU-Anweisung auch Zugriffe auf den von RTOS-UH benutzten Speicherbereich erfolgen. Mit 'SEND' auf diesen Bereich ist das Überschreiben von System- und Anwendersoftware möglich.

Zu dieser Station gehört die Betreuungstask #EDFMN. Damit können - solange genügend Speicher vorhanden ist - beliebig viele benannte Textdateien ähnlich wie auf einem Massenspeicher verwaltet werden. Die Betreuungstask #EDFMN kennt keine Wartephasen, da ja kein physikalisches Gerät in dem Transport verwickelt ist.

Der Text wird im Speicher in untereinander vernetzten Blöcken ('EDTF') abgelegt. Bei der Station ED werden einfache Redundanzen (Wiederholungen von Buchstaben, nachstehende Leerzeichen und ähn.) für die Speicherung des Textes komprimiert, dies geschieht aus Speicherplatzersparnis für den Text.

Die Station EDB bildet den Text in seiner ursprünglichen Form in die Textdatei ab, d.h. daß binäre Daten abgelegt werden können (siehe READ/WRITE).

Der in den ED-Dateien abgelegte Text kann durch Einloggen in den bildschirmorientierten Texteditor (siehe ED-Kommando) verändert werden.

A c h t u n g: Logt man in ein EDB-File ein, kann der Editor gestört werden und die Daten können verändert werden.

Erlaubte Operationen sind:

Read , Write , REWIND , ERASE , RM , FILES , DIR

BEDIENINTERFACE: Name der Station ist ED oder EDB. Die Angabe eines Filenamen (bis 24 Zeichen) ist notwendig, da andernfalls der Ersatzname '-' (set by System) eingesetzt wird. Nur bei 'FILES' oder 'DIR' kann auf einen Filenamen verzichtet werden.

BEISPIEL:

```
ERASE ED:Test , ED:Kopie
REWIND ED:Data
COPY  ... > ED:Program
```

PEARL-PROGRAMM: Systemname ist ED. (bzw. LD/1,1/ für EDB.). Wird kein Filename zugefügt, so wird der am weitesten rechts stehende Nutzername als Filename eingesetzt.

BEISPIEL: (Station ED normale ASCII-Zeichen)

```
MODULE M;
 SYSTEM; POOL: ED.SAVE ->;
 PROBLEM; SPC POOL DATION OUT ALPHIC CONTROL(ALL);
 .....
 AA: TASK;  ....
    OPEN POOL; CALL REWIND (POOL);
    PUT x,x**2 TO POOL BY (2)E(20,10);
    .....
    CLOSE POOL;
 END;
MODEND;
```

BEISPIEL: (Station EDB binäre Daten)

```
MODULE M;
   SYSTEM; DATA: LD/1,1/ ->; /*EDB: STATION*/;
   PROBLEM; SPC DATA DATION OUT ALPHIC CONTROL(ALL);
   SPC WRITE ENTRY GLOBAL;
AB: TASK; DCL SSS(1000) FLOAT;
    OPEN DATA; CALL REWIND(DATA);
    CALL WRITE (DATA,SSS); /*Daten werden binär gespei-
                             chert*/;
    CLOSE DATA;
END;
MODEND;
```

HINWEISE: Wird nach einer REWIND-Operation erneut in den File geschrieben, so wird die Länge des Files bis auf den Stand des Schreibzeigers gekürzt, ggf. früher eingeschriebene Daten (Text) gehen somit verloren.

Falls bei einer Schreiboperation kein Platz für die Anforderung eines weiteren Blockes mehr zur Verfügung steht, so wird eine Fehlermeldung und der Returncode RECLEN=0 (wichtig für Assemblerprogrammierer) abgesetzt. Die zu schreibenden Daten gehen dabei verloren.

Bitte überzeugen Sie sich zunächst (Tabelle am Ende des Schnellkurses) auf welcher Warteschlangennummer Ihre Floppy bzw. ggf. Ihre Winchester implementiert wurde, da Sie diese Information für den PEARL-Compiler benötigen. In den meisten Fällen wird die Diskette auf der LDN No. 3 zu finden sein. Dafür ist im Compiler ein passendes Schlüsselwort vorhanden. (F0 ... F3). Es ist möglich, daß für Winch. und Diskette die gleiche LDN gilt. Die Selektion erfolgt dann über die Drivenumber.

Der Fileaufbau ist hierarchisch, es besteht aber kein Zwang zur Benutzung einer solchen Baumstruktur. Insbesondere können ältere Disketten des nichthierarchischen Filehandlers immer noch einwandfrei gelesen werden.

Die Winchester kann scheinbar in verschiedene Laufwerke aufgeteilt sein. Zu diesen scheinbaren Laufwerken gehören dann auch eigene Directories. Grundsätzlich gibt es keinen Unterschied beim Umgang zwischen Disketten und Winchester.

Studieren Sie bitte in B-II die Kommandos 'FORM','FILES','FREE', 'CF' (evtl. auch für Winch!), 'SYNC', 'REWIND', 'RM'/'ERASE', 'MKDIR', 'RMDIR', 'RETURN' und 'DIR'.

Wichtig ist, daß nicht versehentlich Files geöffnet zurückbleiben oder bei geöffneten Files die Diskette gewechselt wird.

Unsere Floppy/Winch-handler erlauben einen wahlfreien Zugriff von PEARL-Programmen aus mit Hilfe der Einbaufunktionen 'SEEK' und 'SAVEP'.

BEDIENINTERFACE: Stationsname ist Fx (x=0,1,..) für die Disketten und Hx (x=0,1,..) für die Winch.

BEISPIEL:

```
LOAD H0:usr/games/kalaha;

SYNC F1:; CF F1:FORGET; (Bei eiligem Aufbruch)
```

PEARL-PROGRAMM: Der Systemname Fx führt immer auf die LDN 3 !! Also ggf. über Lx (s. A-II 'Lx') adressieren.

BEISPIEL:

```
SYSTEM; Wfile1:LD/5,5/.platzhalter12345<->;
......

SPC  Wfile1 DATION INOUT ALPHIC CONTROL(ALL);
......

OPEN Wfile1 BY IDF('mueller/daten');
CALL REWIND(Wfile1);
PUT x,y TO Wfile1; CALL SAVEP(Wfile1,Pos);
```

Wenn eine Station keinen mnemotechnischen Namen besitzt, so kann sie über das Bedieninterface durch Angabe ihrer Warteschlangennummer adressiert werden. Allerdings erlaubt dieser Zugriff zur Zeit nur die Ansprache des Laufwerkes (Drive) No.'0'. Der PEARL-Compiler gestattet dagegen auch die freie Wahl der Laufwerksnummer. (siehe auch C-I, Sprachumfang PEARL).

BEDIENINTERFACE: Name der Station ist 'Lx'. (x=0,1,..9,A..Z). Eine angegebene pathlist wird an die entsprechende Betreuungstask weitergeleitet. Es kann nur das 'drive' 0 adressiert werden.

BEISPIEL:

```
COPY ED:test co L5:abc/xyz;
```

PEARL-PROGRAMM: Die Station wird über das Schlüsselwort 'LD' angesprochen. Dahinter erfolgt,durch slashes eingerahmt, die Angabe von Warteschlangennummer und drivenumber.

BEISPIEL:

```
SYSTEM;  Flop2:LD/5,2/.xxxxxxxxxxxx<->;
......   /* LDN=5, DRIVE=2  */;

SPC Flop2 DATION INOUT ALPHIC CONTROL(ALL);
......

PUT data1,data2  TO Flop2 BY (2) LIST;
```

HINWEIS: Natürlich können über diesen Weg Warteschlangennummern angegeben werden, zu denen gar keine Betreuungstasks existieren. (Systemintern ist dann kein 'Task-identifier' in der LDN-TID Tabelle eingetragen). In solchen Fällen meldet sich das System beim verantwortlichen User mit >> taskname: WRONG LDN (XIO) und es findet keine Operation statt.

Die parallele Druckerschnittstelle des Rechners wird angesprochen und die Daten werden im Handshakemode interruptgesteuert übertragen. Die Druckerbereitschaft wird nicht explizit geprüft, ist also kein Gerät angeschlossen oder dieses nicht bereit, so bleibt die Betreuungstask #PPORT so lange hängen, bis ein Data-acknowledge empfangen wird (Zustand: #PPORT: SUSP).

BEDIENINTERFACE: Name der Station ist 'PP'. Ein evtl. angegebener Filename bleibt ohne Wirkung. Das Gerät ist dem System nur als Ausgabeeinheit bekannt. Mit Hilfe des 'SD'-Befehles kann das Auto-linefeed den Erfordernissen entsprechend parametriert werden.

BEISPIEL: COPY ED:Test>PP:

PEARL-PROGRAMM: Der Systemname ist PP

BEISPIEL:

```
SYSTEM;
PRINT: PP ->;
.....
PROBLEM;
SPC Print DATION OUT ALPHIC CONTROL(ALL);
.....
PUT message TO Print BY .....;
```

Die Stationen VI (Virtual Input) und VO (Virtual Output) besitzen jeweils eigene prioritätengeordnete Warteschlangen, aber nur eine Betreuungstask #VDATN, die deren Betreuung übernimmt. Alle Ausgabe-Communication-elements, die nach VO geschrieben werden, sowie alle Eingabe-CE's, die von VI gelesen werden, werden in eine betreuungstask-eigene Warteschlange übernommen. Die Übertragung der Daten von den VO-CE's in die VI-CE's erfolgt, sobald auf beiden Seiten CE's vorhanden sind. Die Stationen arbeiten nach dem FIFO-(First In - First Out) Prinzip, d.h. die zuerst erfolgte Ausgabe an eine VO-Datei wird als erste Eingabe von der gleichen VI-Datei gelesen.

VI und VO bilden ein Instrument zur synchronisierten Task-Kommunikation. Erzeuger können ihre Daten zu einem beliebigen Zeitpunkt in die Datenstation VO schreiben und ohne Unterbrechung weiterarbeiten; auch eine Terminierung des Erzeugers führt nicht zum Datenverlust. Verbraucher können jederzeit Daten von VI anfordern, werden aber bis zur Erfüllung der Eingabeanforderung angehalten.

Z.B. können Compiler und Lader über VI und VO verbunden werden, um das Compilat nicht als (speicher- oder floppy-) residente Datei abzulegen.

Bedieninterface: Die Stationen heißen VI: und VO:, der Dateiname kann einen Path enthalten. Es können beliebig viele Dateinamen gleichzeitig verwendet werden.

Beispiel:

```
LOAD VI:Loader; PEARL ...>VO:Loader

RM  VO:Mist  (wenn VO:Mist überflüssig ist)
```

PEARL-Programm: Systemnamen sind VI (nur Eingabe) und VO (nur Ausgabe). Wird kein Filename angegeben, so wird der am weitesten rechts stehende Nutzername verwendet; wird kein Nutzername angegeben, so wird der Filename '-' eingesetzt.

Beispiel:

```
SYSTEM;
   PRODUCE: VO.mypipe ->;
   CONSUME: VI.mypipe <-;
PROBLEM;
   SPC  PRODUCE DATION OUT ALPHIC CONTROL(ALL);
   SPC  CONSUME DATION IN  ALPHIC CONTROL(ALL);
   T1: TASK;
      .........
      PUT ... TO PRODUCE BY ...;
      .........
   END;
   T2: TASK;
      .........
      GET ... FROM CONSUME BY ...;
      .........
   END;
```

Zu dieser Station gehört die Task #XCMMD, die die Betreuung der Ausgabeschlange (prioritätsgeordnet) übernimmt. Mit dem an dieser Station eintreffenden Text wird das Bedieninterface beschickt, die Kommandos werden also auf der Prioritätsebene der Task #XCMMD ausgeführt, sofern nicht Subtasks gebildet werden.
Die an der Station eintreffenden Sätze sollten jeweils mit CR oder LF abgeschlossen werden. Leere Sätze haben keine Wirkung.

BEDIENINTERFACE: Name der Station ist XC. Ein hinzugefügter Filename wird zwar in die Verwaltung übernommen und erscheint beim 'S'-Befehl des Bedieninterfaces, er nimmt aber auf den Ablauf keinen Einfluß.

BEISPIEL:

```
COPY  ...>XC:
```

PEARL-PROGRAMM: Systemname ist XC. Wird kein Filename zugefügt, so wird der am weitesten rechts stehende Nutzername eingesetzt. Wie beim Bedieninterface hat der Filename jedoch keine Wirkung auf den Ablauf.

BEISPIEL:

```
SYSTEM;
BEDIEN: XC;
.........
PROBLEM;
SPC BEDIEN DATION OUT ALPHIC CONTROL(ALL);
.........
TT: TASK;
.........
PUT 'UNLOAD TEST' TO BEDIEN BY A,SKIP;
.........
```

RTOS-UH verwaltet maximal 32 Prozessinterrupts, die eine hohe zeitliche Auflösung ermöglichen, da die Interruptroutine typischerweise unter 100 us durchlaufen werden kann. Fuer einige Implementierungen von RTOS-UH gibt es fest zugeordnete Leitungen zu einzelnen Bits des Interruptbitmusters. Diese Zuordnungen entnehme man den Hardwareunterlagen. Bei allen Systemen später als NUK 2.9 besteht die Möglichkeit, daß der Anwender sich seine eigene 'bitmustererzeugende' Unterbrechungsroutine - etwa für einen speziellen Controller - hinzufügt.

Ein Interrupt wird im RTOS nur wirksam, wenn sein zugehöriges Bit in der 'ENABLE'-Maske des Systemes gesetzt ist. Bei einem Kaltstart von RTOS-UH werden zunächst alle Prozessinterrupts abgeschaltet.

BEDIENINTERFACE: Siehe Anweisungen 'ENABLE', 'DISABLE', 'TRIGGER' und 'WHEN'.

PEARL-PROGRAMM: Systemname ist EV(hexnum8). Das Bitmuster hexnum8 gibt an, auf welche Bits reagiert werden soll.

```
SYSTEM; Limit: EV(00000006);
.......
PROBLEM; SPC Limit INTERRUPT;
.......
TS:TASK; ... WHEN Limit ACTIVATE XYZ ....
```

Sowohl das Interruptbit 00000002 als auch das Bit 00000004 führen jetzt zur Aktivierung von XYZ. (logisches 'oder')

HINWEIS: Mit Hilfe der 'TRIGGER'-Anweisung kann die Wirkung eines äußeren Ereignisses exakt simuliert werden, da systemintern die gleichen Programmteile angestoßen werden.

EINBAU EIGENER PROZESSINTERRUPTS:

Der Nutzer schreibt eine Interruptroutine und versorgt den Interruptvektor sowie die sogenannte 'Malfunction' (s. dazu unbedingt E-VI, Codierung des Interruptprozesses) mit Hilfe einer Task oder des 'GO'-Befehles. Dabei müssen genau die angegebenen Register gerettet werden, da sonst der Rückfallmechanismus zum Absturz führen kann.

Der Anschluss könnte zum Beispiel wie folgt aussehen:

```
            DC.L      0,0        Für RTOS-Lader s. D-III !  *
            DC        $0010      Modulkopf            ''    *
            DC.B      'Prozir'   Name des Modules     ''    *
     *
     IID    EQU       $7FE       Interrupt-identifier       *
     IRVEC  EQU       ????       Vector-link je nach Hardware
     TERV   OPD       $A010      Terminate and vanish       *
     *  ...  Hier Einstieg fuer 'GO'-Befehl                  *
            MOVE.L    =IRP,IRVEC  Einsetzen                 *
            TERV                 'GOTO'-Subtask killen.     *
     *
     *
            DC        IRPE-IRP   Anschluß zu Fehlerrückfall
     IRP    MOVE      IID,-(A7)  Save old Interruptidentifier
            MOVE      =IRVEC,IID For any  malfunction-process
 -->        MOVEM.L   D1/D6/D7/A1,-(A7)  ist vorgeschrieben!!
            ......
            ......    Interruptbitmuster vom Coupler lesen
            ......    und den Interruptrequest des Couplers
            ......    zurueckset zen. (IR-Ursache beseitigen).
            ......    Triggerbitmuster nach D1.L schaffen.
            ......
            MOVEA.L   $80E,A1    Zieladresse innerhalb RTOS
            JMP       (A1)       Zum Systemkern.
     *
     IRPE   MOVEM.L   (A7)+,D1/D6/D7/A1    Reload  registers
            MOVE      (A7)+,IID  Rueckladen des Interrupt-ID
            RTE                  Rueckfallabschluss
```

Durch einmalige Exekution von GO auf den Platz (Ladeadr+$10) wird die neue Prozessinterruptbehandlung aktiviert. Dies ist die platzsparendste Lösung, bequemer wäre es, eine richtige Anschlußtask zu schreiben und diese namentlich zu starten.

WICHTIGER HINWEIS: Nach Abort/Reset wird dieser Anschluß wieder aufgehoben, es muß also ein erneuter 'GO' Befehl abgesetzt werden. Vor dem Entladen des angeschlossenen Interruptcodes muß umgekehrt das alte Link wieder hergestellt werden, eben z.B. mit Abort. Für dauerhafte Sonderbehandlung wird eine Einbettung in den EPROM-Code empfohlen. Siehe dazu Scheibe 14, E-VII. In diesem Fall Ändert sich am Code der Interruptroutine nichts, lediglich der Vektoranschluß wird durch die Scheibe automatisiert.

This page intentionally left blank.

Das Betriebssystem verwaltet das verfügbare RAM vom unteren und oberen Ende aus nach zwei verschiedenen Philosophien.

Anforderungen, die meistens zu längerer Speicherbelegung führen, werden durch eine Suche von unten nach oben mit dem 'first fit'-verfahren befriedigt. Dazu gehören folgende Speichersektionen:

EDTF	Editor-files
MDLE	PEARL- oder Assemblermodule bei Laden ohne Adreßzusatz
TASK	PEARL- oder Assemblertasks beim Laden ohne Adreßzusatz

Anforderungen, die in der Regel nur kurzfristige Belegung vermuten lassen, werden durch eine Suche von oben nach unten, ebenfalls nach dem 'first fit', abgedeckt. Dazu gehören folgende Sektionen:

TWSP	Task-workspace. Speicher für taskeigenene Variable.
PWSP	Prozedur-workspace. Speicher (taskzugehörig!) für die in Unterprogrammen deklarierten Variablen.
CWSP	Communication-workspace. Das interne Botschaftensystem von RTOS-UH bedient sich solcher 'CE' (comm. element).
TASK	Header von Sohnprozessen, wie sie etwa beim 'P', 'COPY' etc. vom Bedieninterpreter geschaffen werden.

Teilweise bedienen sich Sohnprozesse wie Lader und Floppyhandler für ihre Datenhaltung mehrerer Sektionen des Typs PWSP. Wenn das Betriebssystem diesen Platz nicht mehr zur Verfügung stellen kann, so läuft der Sohnprozeß genau wie normale Tasks nicht weiter.

Nach unserer Erfahrung entmischen sich bei diesem Prinzip die Sektionen meistens automatisch auf günstige Weise. Vereinzelt kann es vorkommen, daß man mehrere einzelne freie Sektionen noch zur Verfügung hat, und nur deren Summe ausreichend wäre. Wenn der Platz zwischen EDTF-Blöcken liegt, so kann man sich mit einem Umkopieren mit Hilfe einer Floppy behelfen. Dabei ist aber zu bedenken, daß der Floppyhandler selbst ein ziemlich gefräßiger Geselle ist und u.U. in diesem Grenzfall nicht lauffähig ist.

MERKE: Wenn ein Sohnprozeß nicht startet, trotz richtiger Prioritätslage, oder der Floppyhandler stecken bleibt, so schaue man mit dem 'S'-Kommando nach, ob nicht ganz einfach Speicherplatz fehlt. Der Bedieninterpreter erhält seinen Speicher schon beim Systemurstart und ist daher immer ansprechbar. ABER: Man kann sich bei der Grenzbelegung leicht aus dem System ausblenden, wenn etwa das Kommando 'DIR' oder 'RM' gegeben wird, da in diesem Fall der Bedieninterpreter auf den nie fertig werdenden Floppyhandler bis zum St.Nimmerleinstag wartet! Evtl. über ein anderes Nutzerterminal Platz schaffen oder Aborttaste betätigen.

Praktisch ist aber nur bei den sehr kleinen 68008-Systemen mit nur wenigen kB RAM mit solchen Problemen zu rechnen.

This page intentionally left blank.

Das Betriebssystem RTOS-UH bedient sich einer zentralen Meldetask #ERROR, um Meldungen über Irregularitäten an den für die laufende Task verantwortlichen Nutzer abzusenden. Man sollte dazu wissen, daß diese Task für jeden Nutzer einen kleinen Ringpuffer hat, in dem nur Platz für 6 Meldungen ist. Kommen weitere Fehlermeldungen an, bevor wieder Platz im Puffer ist, so gehen diese verloren. Aus diesem Grund sollte man sich nicht an den rasenden Meldungen, z.B. Division durch Null, erfreuen, sondern daran denken, daß vielleicht wichtige Nachrichten verloren gehen können. #ERROR liegt in der Priorität höher als der Bedieninterpreter, bei manchen Terminals muß man bei rasenden Meldungen (wegen der Blokkierung durch das Terminalprotokoll) daher ziemlich hartnäckig auf die Ctrl A-Taste hacken, um wieder in das System zu kommen.

Jede Meldung über #ERROR beginnt mit dem typischen '>>' am linken Rand der Terminalzeile. Diesem String folgt typischerweise der Name der verursachenden Task, eine Adresse oder weiterer Name, sowie die eigentliche Botschaft. Je nach Schwere des Fehlers wird die Task suspendiert oder kann weiterlaufen.

>> ABCD: 008022 WRONG OP-CODE (TRAP)

Lies: Die Task ABCD ist auf einen unbekannten Befehl gelaufen, der Programmzähler steht jetzt auf 008022. Der falsche Befehl (Speicherfehler?, Feldüberschreitung ohne Tester?) muß also kurz vor dieser Adresse zu finden sein. Wenn die Task ABCD eine PEARL-Task ist und mit der /*+M */-Option übersetzt wurde, so kann man mit dem 'DL'-Kommando nachsehen auf welcher Programmzeile die Task jetzt steht.

>> COPY26: (TERMI)

Lies: Hier gibt es offenbar gar keinen Fehler, sondern nur einen Hinweis, daß der Sohnprozeß COPY26 jetzt seine Arbeit beendet hat.

>> **V200:08FFA2 WRONG ADDRESS (Nur auf Konsolterminal möglich)

Lies: Die Sterne bedeuten, daß es sich keine Task zuordnen läßt, weil der Fehler in einem Interruptprozeß aufgetreten ist. Dabei bedeutet V200, daß der Interruptprozeß über den Exceptionvektor $200 angeschlossen ist. Meistens ist das eine ungute Sache, der Kollaps des Systemes konnte nur durch den integrierten Interruptrückfallmechanismus von RTOS-UH verhindert werden. Der Interruptprozeß hängt also nicht, sondern wurde geordnet abgebrochen.

>> MASTER:MURKS NOT SUSPENDED (CONTINUE)

Lies: Die Task MASTER hat eine CONTINUE-Anweisung für die Task MURKS ausgeführt. Das Betriebssystem stellt aber fest, daß MURKS gar nicht suspendiert ist und die Anweisung somit ohne Wirkung bleiben muß. Keine schlimme Sache, irgendein kleinerer Denkfehler des Programmierers beim Tasking.

This page intentionally left blank.

Einige technische Daten

Maximale Anzahl gleichzeitig aktiver Tasks:	Nicht begrenzt!
Maximale Anzahl von Semaphorvariablen:	Nicht begrenzt!
Maximale Anzahl von Prozeßinterrupts:	32
Reaktionszeit auf Prozeßinterrupt unter PEARL:	kleiner 200 us

Prozeßumschalter ('Dispatcher'):

Rein ereignisgesteuert, trennt keine Sequenzen im Supervisormode. Anstoß durch Flag ($800 neg.) und Exceptionexit.

Umschaltzeit (8 MHz CPU, no waitstates):	besser 200 us
Wechselzyklus A-B-A:	besser 400 us

I/O-System:

Prioritätsgeordnete Warteschlangen mit eigenen Betreuungsprozessen (I/O-Tasks). Ein- und Ausgabe ist ohne Wartephasen des Auftraggebers möglich, obwohl die Information nicht umkopiert wird. Durch einen Trap (XIO) wird nicht (wie sonst üblich) die Ein-/Ausgabe quasi per Unterprogramm erledigt, sondern ein unabhängiger I/O-Prozeß gestartet. Ideal für symmetrische Mehrprozessorsysteme.

Nutzerdatenstationen können permanent oder temporär hinzugefügt werden.

Bedieninterpreter:

In 68000-Maschinencode codiert. Befehlssatz erweiterbar.

PEARL-Compiler:

In virtuellem (VCP-) Code geschrieben. Ca. 500 Zeilen/min.

Lader:

Selbstbindend, kann EPROM-residente Prozeduren einbinden.

Portabilität:

Durch Austausch der Implementierungsscheibe an alle Rechner mit 68000, 68010 oder 68020 Mikroprozessor anpaßbar, sofern im unteren Bereich $0 bis ca. $2000 RAM verfügbar ist. Ab $800 muß dieses RAM im User-Mode adressierbar sein.

Urheberrechte:

Prof. Dr.-Ing. W. Gerth, Institut für Regelungstechnik
Universität Hannover
Appelstr. 11

This page intentionally left blank.

Allgemeines

Das Bedieninterface ist eine wiedereintrittsfeste Prozedur mit unmittelbaren Verbindungen zum Kern des Betriebssystems. Der Nutzer kann diese Prozedur auf zwei verschiedenen Wegen aktivieren:

1. Über die seinem Terminal zugeordnete hochpriorisierte Treibertask, die den Namen #USERx hat.
   Dazu wird das Zeichen 'Control A' auf dem Terminal angeschlagen. Die Treibertask zeigt ihre Bereitschaft durch Ausgabe des Zeichens '*' an und erwartet die Eingabe einer Anweisungszeile. Nach Abarbeitung der Zeile kehrt die Treibertask wieder in ihre Grundstellung zurück.

2. Durch Einschreiben von Text in die Datenstation 'XC'. Die an dieser Datenstation eintreffenden Records werden in einer prioritäts-geordneten Warteschlange aufgefangen und nacheinander dem Bedieninterface zugeführt. Über diesen Weg können z.B. in Hochsprache geschriebene Programme selbst alle Bedieninterface-Kommandos wie ein menschlicher Bediener absetzen. So können z.B. durch zyklische Einplanung einer 'COPY'-Subtask beliebige Befehle automatisch wiederholt werden:

   ```
   COPY.X ED:XC>XC:;T X;ALL 10 SEC X
   ```

Anweisungsformat

In einer Anweisungszeile dürfen keine, eine oder mehrere Anweisungen stehen. Mehrere Anweisungen in einer Zeile müssen durch Semikolon voneinander getrennt werden. Sie werden dann parallel abgearbeitet. Wenn die Anweisungen hintereinander abgearbeitet werden sollen, sind sie durch zwei Minuszeichen zu trennen (s.u.). Leeranweisungen sind auch zwischen Semikolons zulässig. Zwischenräume (Blanks) sind zwischen Schlüsselworten und Parametern erforderlich, wenn andernfalls Mißverständnisse entstehen könnten. Mehrere Parameter in einer Parameterliste können wahlweise durch Blanks oder Kommata getrennt werden. Eine Zeile wird durch das Zeichen 'Carriage-return' beendet.

Beispiele für zulässige Eingaben:

```
DM 5000;;;

SM 1000,1200;S;L;XYZ PRIO 50;TERMINATE ABCD;   LOAD ED:BB

T XYZ; TRACE TEST L344,23;
```

Aufruf über eine Bedientask

Die Zugriffszeit (nach Anschlag von ctrl A) hängt von der dem Terminal zugeorneten #USERx-Task und der aktuellen Systembelastung ab. Die Priorität ist allerdings die zweithöchste - nach der #ERROR-Task - im ganzen System, so daß die Reaktion prompt erfolgen sollte.

A c h t u n g :

Eine eventuell noch 'hängende' Eingabe (z.B. erwartet eine Task eine Eingabe) kann den Zugriff verhindern. Erscheint das Zeichen '*' nicht, daher probeweise 'Carriage-return' eingeben.

Eine Eingabezeile darf max. 128 Zeichen lang sein.

Abbruch der Treibertask durch 'BREAK' möglich - neue Eingabe.

Trotz der hohen Priorität der Treibertask werden andere laufende Aktivitäten des Betriebssystems dadurch praktisch nicht behindert, so daß keinerlei Grund zur Zurückhaltung z.B. während einer laufenden Compilation besteht. Zeitlich aufwendige Operationen werdem vom Bedieninterface durch Erzeugung von Subtasks erledigt und damit auf eine niedrigere Prioritätsebene verlagert (kein Abbruch mit 'BREAK', sondern UNLOAD!).

Aufruf über die Datenstation 'XC'

Jedes ASCII-Record, welches durch COPY oder Programmbefehle in die Ausgabestation 'XC' geschrieben wird, wird genauso behandelt, wie eine vom Nutzer über seine Bedientask abgesetzte Zeile. Damit ist auch das automatische und einplanbare Laden, Entladen, Compilieren etc. möglich.

Solange mit dem 'O'-Kommando nichts anderes vereinbart wird, wird als Ausgabegerät das jeweilige Terminal des Nutzers angenommen. Bei Umschaltung durch das 'O'-Kommando gilt dieses nur für den Rest der Zeile bzw. bis zum nächsten 'O'-Kommando.

Beispiele:

Beginnend um 12:00:00 Uhr soll alle 2 Sekunden der Zustand der Task TRANS aufgelistet werden. Die Zustände sollen in die Datei ED:ZUST geschrieben werden. Um 12:02:50 soll der letzte Eintrag erfolgen:

```
COPY A1:>ED:CMMD        Erstellung der Befehlsdatei

=O ED:ZUST;SHOW TRANS;TERMINATE LIST  steht im File

=(EOT)                 Ctrl-D schließt die File Erstellung

COPY.LIST ED:CMMD>XC:;T LIST

AT 12:00:00 ALL 2 SEC UNTIL 12:02:50 LIST
```

Ein in PEARL geschriebenes Modul (TEST) soll ganz aus der Verwaltung von RTOS-UH verschwinden:

```
MODULE TEST;
SYSTEM;  XC:XC.;
PROBLEM; SPC XC DATION OUT ALPHIC CONTROL(ALL);
....
ABCD:TASK;
....
   PUT 'UNLOAD TEST*' TO XC BY A,SKIP;
   SUSPEND;
END;
....
MODEND;
```

Hinweis: Die Priorität der Betreuungstask für die 'XC'-Warteschlange liegt zwar über allen Anwendertasks, dennoch kann der im letzten Beispiel eingebaute SUSPEND-Befehl evtl. zur Ausführung kommen, weil noch andere Aufträge weiter vorne in der Warteschlange stehen können und die mit dem 'PUT'-Befehl abgesetzten Anweisungen dadurch verzögert werden.

Zeitliche Hintereinanderschaltung von Befehlen

Sollen verschiedene Befehle in zeitlicher Folge hintereinander ausgeführt werden, so sind zwischen den Befehlen zwei Minuszeichen einzugeben.

Beispiel: Es soll ein PEARL-Programm mit dem Namen TASK1 übersetzt, geladen und aktiviert werden.

Eingabe: P--LOAD--ACTIVATE TASK1

Nach dem Übersetzen wird das Programm geladen und gestartet. Die Meldungen des jeweiligen Befehls werden durch den nachfolgenden Befehl unterdrückt.

Tritt während einer Befehlsausführung ein Fehler auf, werden die nachfolgenden Befehle nicht mehr ausgeführt. D.h. wenn während der Kompilierung ein Fehler auftritt, wird das Programm nicht mehr geladen und nicht aktiviert.

Kommandos des Bedieninterfaces

A or '	tasknamelist	Aktiv. von Tasks ggf. mit Angabe der Laufpriorität.
ACTIVATE	''	Wie A.
AFTER	schedule,task	Einplanung zeitverzögert.
ALL	schedule,task	Einpanung,zyklisch.
AS	(subtask,paralist)	Assemblieren durch Subtask.
ASSEM	''	Wie AS.
AT	schedule,task	Einplanung für Zeitpunkt.
C	tasknamelist	Continue suspended Tasks.
CD	(devbez.)pathlist	Workingdirectory einrichten/(erweitern).
CF *)	floppydev.extrainf	Diskettenentnahme aus Laufwerk
CLOCK	--	Uhrzeit+nächste Einplanung ausgeben.
CLOCKSET	time-specif.	Rechneruhr stellen.
CONTINUE	tasknamelist	Wie C.
COPY	(subtask,paralist)	Kopieren und/oder mischen.
DD	device	Parameterbytes anzeigen.
DIR	floppydevlist	Directory auflisten.
DISABLE	eventcode	Interrupt abklemmen.
DL	taskname	Zeilennummer PEARL-Prog. ausgeben.
DM	adr.-exp,(adr or len)	Memory auflisten.
DR	taskname	Display Registers of Task.
ED	(subtask,paralist)	Einloggen Texteditor.
ENABLE	eventcode	Interrupt(s) scharf machen.
ERASE	device-file-list	Löschen von Files.
FILES	devicename	Auflisten der aktiven Files.
FORM *)	'S'or'D',devicename	Floppy formatieren.
FREE *)	floppydevice	Freien Platz ausgeben.
GO	(subtask,paralist)	Sprung auf angegebene Adresse.
L	--	Alle Tasks mit Zuständen auflisten.
LOAD	(subtask,paralist)	Linken und Laden von Modulen in UH-S-Records.
LU	--	Nur Usertasks mit Zuständen auflisten.
MKDIR *)	devbez.pathlist	Directory einrichten.
NOTRACE	taskname	Adressen- und Zeilenüberwachung abstellen.
O	device-file	Ausgabe Bedieninterface neu vereinbaren.
P	(subtask,paralist)	PEARL-Programm kompilieren.
PEARL	(subtask,paralist)	Langform von'P'.
PREVENT	tasknamelist	Einplanungen löschen.
PWD	--	Workingdirectory ausgeben.
RELEASE	sema-adr-list	Semaphore freigeben.
RETURN *)	device-file-list	Files zurückgeben.
REWIND	device-file-list	File zurückspulen.
RM	device-file-list	Wie ERASE.
RMDIR *)	devbez.pathlist	Directory löschen.
S	--	Speicherbelegung ausgeben.
SD	device	Parameterbytes setzen.
SHARE	prio	Timesharing
SH	tasknamelist	Taskzustände angeben.
SHOW	tasknamelist	Langform von 'SH'.
SM	adr.-exp.,value	Speicherzelle(n) setzen.

SU	tasknamelist	Tasks suspendieren.
SUSPEND	tasknamelist	Langform von 'SU'.
SY	--	Wie 'S'.
T	tasknamelist	Tasks beendigen.
TERMINATE	tasknamelist	Langform von 'T'.
TRACE	taskname,adr/line	Adress-/Zeilenüberwachung einschalten.
TRIGGER	eventcode	Interrupt simulieren.
UNLOAD	task-/modulnamelist	Tasks/Module entfernen.
WHEN	eventcode,tasksched.	Task für Interrupt einplanen.

*) Befehl nur bei Systemen mit Diskettenstation vorhanden.

This page intentionally left blank.

Activate Task (by Priority)

SYNTAX: ACTIVATE taskname [PRIO integer3]
'taskname [PRIO integer3]
A taskname [PRIO integer3]
taskname [PRIO integer3]

Beschreibung: Die Task 'taskname' wird mit der Priorität 'integer3' aktiviert.

Ist die Task bereits aktiv, so wird der Aktivierungszähler der Task erhöht (die Anzahl der gepufferten Aktivierungen wird erhöht); andere mögliche Einplanungen der Task werden nicht beeinflußt.

Fehlt der Zusatz 'PRIO', so wird der Defaultwert aus der Task-Definition eingesetzt. Das gleiche gilt, wenn 'integer3' gleich Null ist.

Die Bedienprozedur prüft selbst nicht, ob die Task vorhanden ist. Der Aktivierungsbefehl kann also zur Auslösung eines Fehlersignals durch den Kern von RTOS-UH (hier 'NOT LOADED') führen.

Beispiele: ACTIVATE XYZ PRIO 123;test;'ABCDE

Die Task XYZ wird mit der Priorität 123 aktiviert, während die Tasks test und ABCDE unter Verwendung ihrer Default-Prioritäten aktiviert werden.

Hinweis: Wenn der Taskname gleich einem Bedienkommado ist, kann die Kurzform (nur Taskname) nicht angewendet werden, da zuerst auf ein gültiges Kommando geprüft wird.

Delayed Activation or Continuation

SYNTAX: AFTER duration ACTIVATE taskname [PRIO integer3]
AFTER duration CONTINUE taskname
AFTER duration ALL duration UNTIL clock
ACTIVATE taskname [PRIO integer3]
AFTER duration ALL duration DURING duration
ACTIVATE taskname [PRIO integer3]

Beschreibung: Bestehende Einplanungen für eine Aktivierung (bei ACTIVATE) bzw. zur Fortsetzung (bei CONTINUE) werden gelöscht, und die angegebene Einplanung wird eingetragen.
Wird bei ACTIVATE keine Priorität angegeben, so wird die taskeigene Default-Priorität eingesetzt. Die aktuelle Priorität einer laufenden Task wird jedoch nicht geändert, sondern erst, wenn die Einplanung zur Aktivierung führt.

duration: Textstring: integer5 HRS integer5 MIN
integer5[.integer3] SEC
Bis zu 2 Zeiteinheiten (HRS, MIN, SEC) dürfen fehlen, die Reihenfolge HRS - MIN - SEC muß jedoch eingehalten werden.

ALL: siehe 'ALL'-Schedule. Eine mit DURING angegebene Zeitdauer rechnet ab der ersten Aktivierung.
Diese Kombination ist im DIN-Basic-PEARL nicht erlaubt, aber im RTOS-UH-PEARL wie hier implementiert.

Hinweis: An Stelle der Schlüsselworte 'ACTIVATE' und 'CONTINUE' sind auch deren Kurzformen Hochkomma bzw. 'C' zulässig.
Die angegebene Verzögerungszeit rechnet ab dem Eintritt des nächsten Clock-Ticks. Ist die Anzahl der Millisekunden für die Gesamtverzögerung nicht durch den Abstand der Clock-Ticks (1, 2, 4 oder 8 ms - je nach Implementierung) ohne Rest teilbar, so tritt die Aktivierung bzw. Fortsetzung mit dem ersten Clock-Tick nach Ablauf der Zeitspanne ein.

Beispiele: AFTER 0.25 SEC CONTINUE XYZ

AFTER 10 MIN 5 SEC ALL 1 SEC ACTIVATE XYZ PRIO 70

AFTER 5HRS59MIN22.55SEC ACTIVATE XYZ

SYNTAX: ALL duration ACTIVATE taskname [PRIO integer3]
ALL duration UNTIL clock ACTIVATE taskname
[PRIO integer3]
ALL duration DURING duration ACTIVATE taskname
[PRIO integer3]

Beschreibung: Es wird eine zyklische Einplanung für die angegebene Task definiert und evtl. bestehende zeitliche und ereignisgekoppelte (WHEN) Einplanungen zur Aktivierung gelöscht. Die erste Aktivierung erfolgt mit dem nächsten Clock-Tick und wiederholt sich - ggf. bis zur Endzeit - von da an zyklisch.

Wird die Priorität (3-stellige Ganzzahl) nicht angegeben, so wird die taskeigene Priorität genommen.

duration: ist vom Typ: integer5 HRS integer5 MIN
integer5.[integer3] SEC,

dabei ist 'integer5' eine maximal 5-stellige Ganzzahl und 'integer3' ein max. 3-stelliger Dezimalbruch.

Es dürfen bis zu 2 Zeiteinheiten (HRS, MIN, SEC) weggelassen werden, die Reihenfolge HRS - MIN - SEC muß jedoch stets eingehalten werden.

clock: ist vom Typ:
integer2:integer2:integer2[.integer3]

'integer2' ist eine 1 bis 2-stellige Dezimalzahl und 'integer3' ein max. 3-stelliger Dezimalbruch.

Hinweis: An Stelle des 'ACTIVATE' ist auch die Kurzform mit Hochkomma vor dem Tasknamen zulässig.

Zeitdauer und Uhrzeit werden intern als Vielfache von Millisekunden gerechnet. Je nach Implementierung werden jedoch Clock-Ticks von 1, 2, 4 oder 8 ms als Interruptbasis benutzt. Ist der Zyklus nicht ohne Rest durch diese Basis teilbar, so werden die Zeitintervalle länger oder kürzer, aber im Mittel richtig, realisiert.

Beispiele:
```
ALL 0.02 SEC ACTIVATE XYZ PRIO 30
ALL 13 HRS 2.005 SEC 'XYZ
ALL 2 MIN UNTIL 13:05:55.66 ACTIVATE XYZ
ALL 7000 SEC DURING 14000 SEC ACTIVATE ABC
```

(ABC wird 3 mal aktiviert - sofort, nach 7000 sec und nach 14000 sec)

Generate Subtask and Assemble Program

SYNTAX: AS.subtaskname [PRIO integer3] [size-spec]
[parameterlist]
AS [PRIO integer3] [size-spec] [parameterlist]
ASSEM.subtaskname [PRIO integer3] [size-spec]
[parameterlist]
ASSEM [PRIO integer3] [size-spec] [parameterlist]

Beschreibung: Das Bedieninterface generiert eine flüchtige, eigenständige Task mit vom Nutzer vorgegebenem Namen oder, falls in der zweiten Form benutzt, einem Systemnamen ASMBxx. Für xx wird eine zweistellige Hexzahl mit automatischer Weiterschaltung eingesetzt. Die Priorität dieser flüchtigen Subtask kann vorgegeben oder dem System überlassen werden (Default: 20).

Ebenso kann der dynamische Arbeitsspeicher des so erzeugten 2-Pass-RTOS-UH-Assemblers vorgewählt oder die vom System standardmäßig gewählte Größe von 5 kB benutzt werden. Wenn das Feld 'size-spec' benutzt wird, so ist dies nur bis maximal SZ=7FFE (32 kB) sinnvoll.

Da der Assembler wiedereintrittsfest codiert ist, können, solange der Speicherplatz reicht, beliebig viele Programme gleichzeitig assembliert werden, wobei stets derselbe, im ROM gespeicherte Assembler benutzt wird.

parameterlist Es werden die Elemente SI (Source Input), LO (List Output), CO (Code Output) und SC (SCratch-pad) akzeptiert. Die Reihenfolge ist bedeutungslos, die Liste darf auch leer sein. Für fehlende Angaben werden die 'Default-werte' des Systems eingesetzt (SI=ED:SI, LO=A1:, CO=ED:SR, SC=ED:SC).

Wird ein Parameter auf den Wert NO gesetzt, so gelten dennoch davor oder dahinter gemachte Vereinbarungen bzw. die Defaultwerte, wenn der Assembler das Gerät im Ausnahmefall benötigt.

Das SCratch-pad wird nur benötigt, wenn das Gerät für SI nicht rückspulbar ist (z.B. VI:, A2:). Ist das Gerät für SI rückspulbar, so erfolgt sowohl bei SC=NO als auch bei fehlendem SC-Parameter zweimaliges Lesen der SI-Datei ohne SCratch-pad-benutzung. Wird beim Aufruf des Assemblers nur eine Datei angegeben, so wird diese als SI-Datei betrachtet.

Beispiele: AS.T PRIO 40 SZ 4000 F1:QUELLE>A2: LO NO

Subtaskname ist T .Nur fehlerhafte Zeilen mit Fehlerbeschreibung über A1: ausgeben - keine Liste. Ausgabe der S-Records über A2:. Da SI als Floppy-Datei rückspulbar und SC nicht angegeben ist, wird der File 'QUELLE' auf F1: zweimal gelesen und kein SCratch-pad benutzt.

ASSEM

Aufruf ASMBxx mit Default-werten.

ASSEM.X A2:>NO LO A1: SC F1.BX

Subtaskname ist X. Nur Syntaxprüfung der über A2: eingegebenen Programmzeilen, Ausgabe der Liste über A1:. S-Records werden nicht erzeugt. Der Eingabetext wird auf der Datei BX auf Floppy F1: zwischengespeichert (und ist dort später verfügbar).

Hinweis: Soweit rückspulbare Files benutzt werden, erfolgt automatisch zu Beginn eine ATTACH +REWIND- und zum Abschluß eine RETURN-Operation.

Die generierte Subtask verschwindet nach Abschluß der Assemblierung vollständig aus dem System.

Weitere Erläuterungen zum Assembler finden Sie in Kapitel D-I ff.

At given time activate or continue

```
SYNTAX: AT clock ACTIVATE taskname [ PRIO integer3 ]
        AT clock CONTINUE taskname
        AT clock ALL duration UNTIL clock ACTIVATE taskname
                                        [ PRIO integer3 ]
        AT clock ALL duration DURING duration ACTIVATE
                                  taskname [ PRIO integer3 ]
        AT clock ALL duration ACTIVATE taskname
                                        [ PRIO integer3 ]
```

Beschreibung: Siehe 'AFTER', die Einplanung wird jedoch nicht nach einer Verzögerungszeit, sondern zum angegebenen Zeitpunkt wirksam.

clock: integer2:integer2:integer2[.integer3]

integer2/3 sind max. 2- bzw. 3-stellige Dezimalzahlen

ALL: siehe 'ALL'. Die Kombination AT ... ALL ist im DIN-Basic-PEARL nicht erlaubt, jedoch im Compiler wie hier implementiert.

Beispiele: AT 0:10:00 CONTINUE test

AT 13:00:0 ALL 2 SEC UNTIL 14:0:0 ACTIVATE XYZ PRIO 20

Hinweis: Ist die angegebene Uhrzeit kleiner als die Istzeit, so wird der Wert von clock um 24 Stunden inkrementiert.

Change Directory

SYNTAX: CD device:pathlist

Beschreibung: Der Befehl Change Directory erlaubt es für jeden am System tätigen Nutzer ein Working Directory zu vereinbaren. Das Working Directory wird bei allen Befehlen, die mit Devices und Files arbeiten, eingesetzt, es sei denn, bei dem Befehl wird ein Device, gefolgt von einem Doppelpunkt, angegeben (Weg über die Root-Ebene).

Jeder Nutzer kann sich also sein individuelles Working Directory einrichten und wird damit bei der Verwaltung hierachisch organisierter Daten bestens unterstützt.

device: Ein Device ist eine Datenstation des Systems, z.B ED:, F0: oder A1:. Wichtig ist der Doppelpunkt, er zeigt, daß es sich um ein Device handelt und nicht um eine Datei.

pathlist: Die Pathlist besteht aus bis zu 7 Buchstaben oder Zahlen, gefolgt von einem '/' usw. Sie muß mit einem Buchstaben beginnen. Endet die Pathlist mit einem Subdirectory, muß der '/' am Ende weggelassen werden.

Beispiele:

CD ED:

Das System antwortet mit

WD= ED:-

Damit ist das Working Directory ED: eingerichtet worden und ein COPY-Befehl in eine ED-Datei sieht jetzt so aus:

COPY F0:TEST>TEST

CD H0:PROG/alt

Es wird das Working Directory PROG/alt auf der Festplatte H0: eingerichtet. Um jetzt eine Datei zu compilieren wird folgendes Kommando abgesetzt:

P MESS>ED:TESTSR LO NO

Das Programm H0:PROG/alt/MESS wird übersetzt und die S-Records werden in die Datei ED:TESTSR geschrieben.

CD MIST

Antwort:

WD= H0:PROG/alt/MIST

Hier wurde das Working Directory 'verlängert', da beim Befehl kein device angegeben war.

Mit CD NO wird das Working Directory gelöscht.

Die Länge der gesamten pathlist ist auf 24 Zeichen beschränkt.

Change Floppy

SYNTAX: CF floppydevice:extrainfo
CF floppydevice

Beschreibung: Es wird dem System mitgeteilt, daß die angegebene Diskette entnommen werden soll oder daß (bei Systemen mit Klappenabfrage) die irrtümlich entnommene Diskette wieder eingelegt wurde. Ausserdem kann durch einen 'FORGET'-mode der Handler gewaltsam von der Diskette getrennt werden. (Software-reset).

'extrainfo': Stepping-rate einstellen: CF Fx:Vy (y=0,1,2,3). Bei Systemen mit direkt gesteuertem FD-controller kann damit die Kopfgeschwindigkeit eingestellt werden. (0=schnellster Mode) Muß vor dem Einlegen der Disketten gegeben werden, gilt für alle Drives, x ist also ohne Bedeutung.

'FORGET'-Mode: CF Fx:FORGET;
'RECALL'-Mode: CF Fx:RECALL;

Bei Systemen mit Klappenabfragemöglichkeit kann entschieden werden, was nach irrtümlich entnommener Diskette geschehen soll. Beim Wiedereinlegen (RECALL) wird die noch im Speicher vorhandene Verwaltung weiterbenutzt, wird eine falsche Diskette eingelegt, so wird diese zerstört. RECALL muß daher nach Einlegen zweimal gegeben werden.

CF F1:RECALL;DIR F1;CF F1:RECALL

Beachte: Man sollte sich für das Montieren der Disketten die Benutzung des CF-Befehles zur Regel machen. Die Systemwarnung soll dann zum Retten noch geöffneter Files animieren. Der FORGET-Mode ist als Softwarereset auch ohne Klappenabfrage sinnvoll einsetzbar. (Mit entsprechender Vorsicht).

Fehler: Kein Device angegeben oder Device ist keine Diskette.

>>>...:Fx DIRECTORY ACTIVE. (Files noch offen).

Beispiele: CF F1:; cf f0:; CF F0:V2;

CF F1:FORGET; (Vorsicht!!)

CF F0:RECALL; (Nur bei Systemen mit Klappentest).

Inspect computer-clock

SYNTAX: CLOCK

Beschreibung: Es wird der aktuelle Stand der Rechneruhr ausgegeben. Außerdem wir die Uhrzeit des zeitlich nächstfolgenden Einplanungszeitpunktes hinzugefügt.

Selbst wenn im System überhaupt keine zeitlichen Einplanungen vereinbart wurden, wird ein nächster Einplanungstermin angegeben. Dies ist die 'Geisterstunde', in der die zentrale Rückstellung der Uhr und aller Planungszeitpunkte um 24 Stunden erfolgt.

Beispiele: CLOCK Ausgabe: 05:24:59 NEXT SCHED 10:55:30

CLOCK Ausgabe: 22:01:12 NEXT SCHED 24:00:00

Die Ausgabe der Geisterstunde im letzten Beispiel zeigt an, daß für den laufenden Tag keine Zeiteinplanungen zur Aktivierung oder Fortsetzung mehr vorliegen.

Hinweis: Zeigt die Rechneruhr eine deutliche Tendenz zum Nachgehen, die nicht auf Ungenauigkeiten des Quarzoszillators beruhen können, so ist das Betriebssystem überlastet.

Eventuell kann der Austausch des Betriebssystems gegen eines mit größerem Abstand der Clock-Ticks oder der Austausch des Prozessors gegen eine schnellere Version erforderlich sein.

Eine Überlastung, die sich auf die Ganggenauigkeit der Uhr auswirkt, wird in der Regel durch eine zu große Anzahl von Tasks mit Einplanungen verursacht. Besonders ungünstig ist es, wenn sich Planungszeitpunkte verschiedener Tasks in großer Zahl in einem einzigen Clock-Tick treffen.

Set Computer-clock to given time

SYNTAX: CLOCKSET clock

Beschreibung: Die Rechneruhr wird auf die angegebene Uhrzeit gestellt. Es empfiehlt sich, vorher alle zeitlichen Einplanungen zu löschen, da bei Vorrücken der Uhrzeit u.U. eine große Zahl von Aktivierungen bzw. Fortsetzungen sofort und gleichzeitig fällig werden können.

clock: integer2:integer2:integer2[.integer3]

'integer2/3' sind max. 2- bzw. 3-stelige Ganzzahlen.

Beispiel: CLOCKSET 13:00:00

CLOCKSET 0:0:10.5

Hinweis: Es wird nur die Software-Uhr Ihres Rechners gestellt. Das Stellen der Hardware-Uhr kann implementationsabhängig auch erfolgen.

Continue Task

SYNTAX: CONTINUE taskname
C taskname

Beschreibung: Die angegebene Task wird aus dem Zustand 'suspended' in den Laufzustand gebracht.

Hinweise: Die Bedienprozedur prüft weder, ob die Task überhaupt vorhanden ist noch ob sie - wie vorgeschrieben - im suspendierten Zustand ist. Der Auftrag wird nur syntaktisch geprüft und an das Betriebssystem abgesetzt. Von dort erfolgt ggf. die Auslösung eines Fehlersignals (hier Meldung ... NOT SUSPENDED, ... NOT LOADED).

Beispiele: CONTINUE ABCD; C XYZ; CONTINUE E

Generate Subtask and Copy or Update

SYNTAX: COPY.subtaskname [PRIO integer3] [size-spec] parameterlist
COPY [PRIO integer3] [size-spec] parameterlist

Beschreibung: Das Bedieninterface generiert eine flüchtige eigenständige Task mit vom Nutzer vorgegebenem Namen (1. Form) oder einem vom System erzeugten Namen COPYxx. Für xx wird eine zweistellige Hexzahl mit automatischer Weiterschaltung eingesetzt.

Die Priorität dieser flüchtige Subtask kann vorgegeben oder dem System überlassen werden (Defaultwert ist 20).

Die Benutzung des Zusatzes 'size-spec' ist für den normalen Anwendungsfall nicht sinnvoll.

Da der im Betriebssystem liegende Programmcode wiedereintrittsfest ist, können - solange der Speicher für den ca. 200 Byte großen Task-header reicht - beliebig viele unterschiedliche COPY-Befehle abgesetzt werden, die im Multitasking parallel bearbeitet werden.

parameterlist Es gelten die Parameter

SI (Source-Input)

CO (Copy/Corrected Output)

SC (Source Correction/Command)

Fehlen SI oder CO, so werden die Defaultwerte des Systems eingesetzt (SI=A1:, CO=ED:SI), was normalerweise nicht sinnvoll sein dürfte.
Es gibt - abhängig davon, ob SC angegeben wurde oder nicht - zwei verschiedene Betriebsfälle:

A.) SC fehlt oder SC=NO

Der bei SI angegebene File wird vollständig auf den bei CO angegebenen File übertragen. Es werden max. 128 Zeichen zu je einem 'record' zusammengefaßt und ggf. - je nach Eigenschaft der CO-Datei - nach Ergänzung eines Zeichens 'linefeed' auf die Ausgabedatei übertragen. Als Ende eines 'record' auf der SI-Seite gelten auch die Zeichen 'linefeed', 'carriage-return' und 'EOT'. Die Übertragung wird beendet, wenn ein 'record' nur aus dem Zeichen 'EOT' besteht oder das Gerät der SI-Datei eine End-of-file-Bedingung festgestellt hat. Bei Übertragungsfehlern wird die Subtask vorzeitig beendet und verschwindet nach Ausgabe entsprechender Meldung über das Terminal. Auch die normale Beendigung wird durch ..name (TERMI) angezeigt.

B.) SC wurde einer Eingabedatei zugeordnet.

Es werden die Zeilen aus den Quellen SI und SC gemischt und als 'records' auf die CO-Datei übertragen. Die Anweisungen zum Mischen und - bei rückspulbarer SI-Datei - auch zum Umschichten der aus SI stammenden Zeilen werden als Kommandos ebenfalls über das SC-Gerät eingegeben. Als Unterscheidungsmerkmal zwischen Kommandos und einzumischendem Text auf dem SC-Gerät dient das Zeichen am Anfang einer Zeile. Steht dort das Zeichen '+', so wird die Zeile als Kommando interpretiert. Mit '+' beginnende Zeilen können also nicht eingemischt werden.

+33-455 : Übertrage die Zeilen no. 33 bis einschl. 455 aus dem SI-File in den CO-File. Anschliessend können beliebig viele Zeilen (die nicht mit '+' beginnen) eingegeben werden, die direkt nach CO übertragen werden.

+755 : Übertrage nur die Zeile no. 755 aus dem SI-File in den CO-File.

+855-840 : Unzulässig, die zweite Zeilennummer darf niemals kleiner als die erste sein. Es wird die 'Kommando-Fehler'-Kondition (s.u.) angenommen.

+399-402 : Die Rückkehr vor oder auf die letzte bereits aus dem SI-File übertragene Zeile ist nur bei rückspulbarem SI-File erlaubt, sonst wird die 'Kommando-Fehler'-Kondition angenommen (s.u.). Nach dem Rückspulen wird die SI-Datei durch Lesen von Anfang an auf die angegebene Stelle positioniert und die Übertragung wie gewünscht durchgeführt.

Beispiele: COPY.Z PRIO 16 F0:ALT>F1:NEU

Subtaskname ist Z. Mit Priorität 16 wird der File 'ALT' von Floppy 0 in den File 'NEU' der Floppy 1 kopiert, bis der File 'ALT' an sein Ende oder eine mit 'EOT' beginnende Zeile gekommen ist.

COPY F1:Quelle>F0:Ziel SC A1:

Subtaskname und Priorität werden vom System gewählt. Es werden Kommandos zum Mischen über das Gerät A1: erwartet.

```
Annahme SI-Inhalt:      AAAAAAAAA   (1. Zeile)
                        BBBBB       (2. Zeile)
                        CCC         (3. Zeile)

Angenommene SC-Zeilen:  +2-3
                        KKKK
                        +1      (SI rückspulbar)
                        JJJJ
                        +2
                        'EOT'

Ergebnisfile:           BBBBB
                        CCC
                        KKKK
                        AAAAAAAAA
                        JJJJ
                        BBBBB
                        'EOT'
```

Der Mischvorgang wird beendet, sobald entweder vom SI- oder vom SC-File eine Endebedingung ('EOT' oder End-of-file) erkannt wird, also auch dann, wenn eine Zeile hinter der letzten auf SI vorhandenen Zeile adressiert wird.

Kommando-Fehler: Wird ein falsches Kommando erkannt (Adressierung einer Zeile hinter der letzten vorhandenen ist <u>kein</u> Fehler), so wird die Meldung ...WRONG COMMAND ausgegeben und die Task suspendiert. Nach der Fortsetzung wird das Kommando erneut erwartet.

Hinweise: Floppy-Files werden automatisch mit ATTACH+REWIND eröffnet bzw. neu eingerichtet und zum Abschluß mit RETURN zurückgegeben.

Show Date

SYNTAX: DATE

Beschreibung: Es wird das aktuelle Datum der RTOS-UH-Datumszelle ausgegeben. Wurde diese Zelle noch nicht gesetzt (s. Befehl DATESET), so wird eine Folge von Minus-Zeichen ausgegeben.

Beispiel: DATE
Ausgabe: 01-01-1987

DATE
Ausgabe: ----------
(Datumszelle war noch nicht gesetzt)

Set Computer-Date

SYNTAX: DATESET date

Beschreibung: Die Datumzelle von RTOS-UH wird gesetz.

date: Eine Zeichenkette mit dem Aufbau tt-mm-jjjj
tt: Tag
mm: Monat
jjjj: Jahr (zwischen 1984 und 2162)

Fehler: Es wird geprüft, ob Anzahl Tage/Monat und Anzahl Monate/Jahr zulässig ist. Im Fehlerfall erscheint die Meldung
>>USERx: Date WRONG.

Beispiel: DATESET 29-02-1988

Display device-parameters.

SYNTAX: DD device;

Beschreibung: Für die angegebene Station wird der Inhalt der aktuellen Parameterbytes ausgegeben. Die Bedeutung der einzelnen Bits entnehmen Sie bitte der Beschreibung des 'SD'-Kommandos. Das Kommando wird intern über die 'DM'-Funktion bearbeitet. Dadurch werden mehr Bytes ausgegeben als für die angegebene Station signifikant sind.

Bsp: DD A1:
Ausgabe Adresse des Parameterfeldes und Inhalt.

xxxxxxxx: 3300 zzzz zzzz zzzz zzzz zzzz zzzz zzzz

Lies: LF nach CR ergänzen ($20), dialogfähig ($10).Ausgabe möglich ($02), Eingabe möglich ($01).

Bsp: DD B2:

xxxxxxxx: 3301 zzzz zzzz zzzz zzzz zzzz zzzz zzzz

Lies: Erstes Byte wie im obigen Beispiel. Im zweiten Byte: Editorsteuerung 'Cursor by esc-sequence' durch $01.

Hinweis: Wenn (mit CD) ein Workingdirektory vereinbart wurde und dem Devicenamen kein Doppelpunkt (vorläufig auch noch Punkt möglich) nachgestellt wurde, so gilt die DD-operation für das mit CD-fixierte Gerät ('LDN'), also nach CD ED: und DD B2 z.b. für alle ED-files!!

Directory listing

SYNTAX: DIR floppy/winch/ed-pathlist;

Beschreibung: Die angegebene pathlist muß auf ein Directory zeigen. Dies kann entweder durch Angabe des Gerätebezeichners (ED:, Fx:, Hx:, etc.) geschehen oder durch Pfadgebung zu einem Unterinhaltsverzeichnis. Die zur entsprechenden LDN (Warteschlangennummer) gehörende I/O-Task fertigt auf diesen Befehl hin eine Liste der zum bezeichneten Inhaltsverzeichnis gehörenden Files an.

Da der ED-Filehandler nur pseudohierarchisch arbeitet, erhält man bei ihm eine Liste aller über den angegebenen Pfad erreichbaren Files.

Fehler: Wenn die Geräteeigenschaft kein DIR-Kommando zuläßt (siehe 'SD'-Befehl), so wird der DIR-Befehl mit 'WRONG COMMAND' abgewiesen.

Aus der angesprochenen I/O-task sind die verschiedenen Fehlermeldungen möglich, z.B:

```
........ DRIVE_NOT_READY.            (Floppy/Winch)
........ TRACK000_NOT_FOUND.                 ''
........ DIRECTORY OVERFLOW  (Nicht UH-formatiert)
```

Beispiele:

```
DIR F0:,F1:,H1:USR/MUELLER;
DIR ED:MAIER/QUELLFILES;
DIR;  (es war mit CD ein Workingdirectory fixiert.)
O ED:FILELIST;DIR F0:;
```

Hinweis: Mit dem 'O'-Befehl kann die Liste in einen beliebigen File gelenkt werden, ausgenommen einen solchen, der zur LDN der mit DIR angesprochenen I/O- Task gehört. In diesem Fall bleiben nämlich systembedingt etliche CE-Sektionen (Besitzer: RTOS) bis zum nächsten ABORT in der Verwaltung herumliegen. Das ist nicht gefährlich, vermindert aber so nach und nach den verfügbaren Speicher.

Also bitte nicht: O ED:ABC;DIR ED:KRUEGER;

Statt dessen: Zwischenschaltung der virtuellen Datenstation und eines COPY-Befehles. Am Ende bitte von Hand ein EOT (Ctrl D) durch die VI/VO-Station schicken oder Pipe mit RM klären und COPY mit Unload entfernen.

```
COPY VI:dirlink > ED:ABC;
O VO:dirlink; DIR ED.KRUEGER;
COPY > VO:dirlink;  (dann EOT anschlagen)
```

Disable Processinterrupt

SYNTAX: DISABLE EV hexnum8

Beschreibung: Von dem Bitmuster 'hexnum8', das bei Eingabe von weniger als 8 Hexziffern durch Ergänzen führender Nullen gebildet wird, wird das 1-er-Komplement gebildet. Dieses wird logisch 'UND' mit der Event-enable-maske des Systems verknüpft und das Ergebnis nach dorthin zurückgeschrieben.
Dadurch werden die durch Einsen in 'hexnum8' gekennzeichnete Prozeß-Interrupts gesperrt.

BEISPIEL:

```
DISABLE EV 1

DISABLE EV FFFFFFFF (alle Prozeß-IR gesperrt)

DISABLE EV 0        (unsinnig, ohne Wirkung)
```

Hinweis: Die Anweisung entspricht der gleichnamigen PEARL-Operation.

Display Line-number of Task

SYNTAX: DL taskname (Nur Hochsprach-Tasks)

Beschreibung: Es wird der Inhalt der Zeilenregisterzelle der angegebenen Task ausgegeben. Die Anweisung ist nur sinnvoll, wenn die angegebene Task mit der Zeilenmarkiereroption übersetzt wurde und zum Zeitpunkt des DL-Kommandos Workspace besitzt.

Beispiel: DL REGEL

Hinweis: Es wird die letzte r e g i s t r i e r t e Zeilennummer ausgegeben. Wenn nur Teile des Programms mit der Markeroption übersetzt wurden, kann es sein, daß die Zeilennummer auf einen längst verlassenen Programmpfad zeigt.

Wird der Wert '0000' ausgegeben, so wurde noch keine Zeilennummer registriert.

Bei nicht hochsprachkodierten Tasks wird auch eine scheinbare Zeilennummer ausgegeben, diese stellt jedoch nur einen Zufallswert dar.

Als Fehlermeldungen können >>... NOT LOADED oder >>...NOT ACTIVE erfolgen.

Display Memory

SYNTAX: DM hex-add-expression (Fall A)
DM hex-add-expression1, hex-add-expression2 (Fall B)
DM hex-add-expression1 hex-add-expression2 (Fall B)

Beschreibung: Fall A. Der Wert von 'hex-add-expression' wird auf die nächstkleinere oder gleiche gerade Zahl abgerundet und als Startadresse für die Ausgabe benutzt. Mit dieser Startadresse beginnend werden die Hexadezimalwerte der folgenden 8 Worte aus dem Speicher mit ihren ASCII-Werten ausgegeben. Steuerzeichen werden durch Punkte dargestellt.

Fall B. Der Wert von hex-add-expression1 wird wie im Fall A als Startadresse gewertet. Ist der (ebenfalls auf geraden Wert gerundete) Wert von 'hex-add-ex-pression2' kleiner als die Startadresse, so wird er als Anzahl der (mindestens) auszugebenden Bytes gewertet; ist er größer oder gleich der Startadresse, so werden alle Speicherzellen bis mindestens 'hex-add-expression2' aufgelistet. In jedem Fall werden ganze Blöcke von 8 Worten (16 Bytes) in jeder Zeile aufgelistet.

hex-add-expression: Eine Folge von maximal 8-stelligen Hexadezimalzahlen, die durch '+' Zeichen miteinander verbunden sind. Damit soll dem Anwender in erster Linie das mühselige Addieren z.B. von Ladeadressen und relativem Abstand erspart werden.

Beispiele:

DM 5000	(Ausgabe der Bytes 5000 ... 500F)
DM 610+20,100	(Ausgabe der Bytes 630 ... 72F)
DM 37FF 3901	(Ausgabe der Bytes 37FE ... 390D)

Hinweis: Der Zugriff auf die Speicherzellen erfolgt im Usermode des Prozessors. Dadurch ist es bei manchen Rechnern nicht möglich, sich alle Speicherstellen anzusehen, da die Hardware einen Buserror auslöst. Mit 'BREAK' kann das Kommando abgebrochen werden.

Display Registers of Task

SYNTAX: DR taskname

Beschreibung: Die angegebene Task wird in der Speicherverwaltung gesucht. Ist sie dort nicht vorhanden, so erfolgt Meldung ... NOT LOADED. Ist die Task zwar vorhanden, aber im Zustand DORM, so erfolgt Meldung ... NOT ACTIVE.

Anschließend werden die Register der Task ausgegeben. Die Ausgabe erfolgt in spartanischer Schlichtheit, da die Anweisung ohnehin nur für Bitmuster-Freaks oder Assembler-Programmierer interessant ist:

Adr:	A7 (US)	A7 (SS)	D0	D1
Adr:	D2	D3	D4	D5
Adr:	D6	D7	A0	A1
Adr:	A2	A3	A4	A5
Adr:	A6	St5/4	St3/2	St1/0

Mit St sind die letzten 6 Worte des System-Stacks gemeint. So ist z. B. St1/0 die Adresse hinter einer TRAP-Instruktion, wenn diese auf Taskebene ausgeführt wurde.

Beispiel: DR XVY

Hinweis: Bei suspendierten Tasks können die Registerinhalte mit Hilfe des 'SM'-Kommandos verändert werden, wenn man die bei DR angegebenen Adressen (s. linker Rand) der Registerablageplätze verwendet.

Edit a File

SYNTAX: ED.subtaskname [PRIO integer3] [parameterlist] (Form 1)
ED [PRIO integer3] [parameterlist] (Form 2)

Beschreibung: Es wird eine flüchtige unabhängige Task mit vom Nutzer (1.Form) oder vom System (2.Form) vorgegebenem Namen erzeugt und gestartet. Ein vom System (2.Form) generierter Name hat den Aufbau EDITxx. Die Standardpriorität für den Editor ist 15 (0F hexadezimal).
Der Editor ist reentrant, es kann also mit dem gleichen Code (auch ROM-Resident) auf mehreren Terminals gleichzeitig gearbeitet werden. (Es müssen nur unterschiedliche Dateien editiert werden).

parameterlist: Es werden die Parameter SC (Scratch-Datei auf der der Editor arbeiten soll) und SI (Source-Input, Port über den der Nutzer Eingaben tätigt) akzeptiert.

SC: muß eine ED.datei der Form ED.xyz sein, sonst wird der Editor sofort mit der Meldung .WRONG LDN (MODE) abgebrochen und verschwindet aus dem System.
SI: muß ein Sichtgerät sein, welches im System als 'dialogfähig' bekannt ist (siehe SD, DD -Befehl), sonst erfolgt ein Abbruch mit der Meldung wie bei falschem SC-Paramter. Die Angabe von SC kann entfallen. Die folgenden Aufrufe sind äquivalent:

ED ED:name bzw. ED SC ED:name und ED SC=ED:name

Bei fehlenden Parametern wird SC=ED:SIx und SI=Ay: eingesetzt, wobei x die LDN der Schnittstelle des Nutzer ist.

Sichtgerät: Es ist jedes Terminal geeignet, welches folgende Bedingungen erfüllt: (Typ: TELEVIDEO TVI 925, 950 und kompatible)

- 24 oder 25 Zeilen à 80 Zeichen.
- Automatischer UP-Scroll bei LF in der untersten Bildschirmzeile.
- Autowraparound = autom. CR+LF nach Anschlag des 80. Zeichens einer Zeile. (No Wrap parametrierbar)
- Kein automatischer LF nach Erhalt eines CR.
- Cursorsteuerung über $0A,$0B,$0C,$08,$16. (Umparametrierung auf ESC-Sequezen möglich)
- Terminal wie bei Bedientask im Full-Duplexmode.

Editorsteuerung: Nach dem Start wird die erste Seite des Files aufgeblättert. Existierte der File vorher nicht, so wird er mit der Zeile 'File was installed by ED' als einziger Information neu im System eingerichtet. Der Editor arbeitet ohne Zwischendatei direkt auf dem angegebenen ED-File, Änderungen sind daher von sofortiger Wirkung auf die Datei.

Die Veränderung einzelner Zeichen erfolgt durch Anfahren der Position auf dem Bildschirm und Eingabe eines neuen Zeichens. (Replace-Mode). Der Editor arbeitet grundsätzlich im Replace-Mode.

Sonderfunktionen werden durch Anschlag des Zeichens 'ESC' angewählt, wobei der nachfolgende Buchstabe die Operation bezeichnet. Die Operationsbezeichner wurden im Sinne einer leichten Merkbarkeit gewählt (siehe folgende Operationstafeln).

Es sollten nur Zeichen eingegeben werden, die einen ASCII-Wert von größer $1F haben. Enthält die Datei Zeichen, deren Wert kleiner als $1F ist (z.B. durch einen COPY-Befehl erzeugt), so werden diese mit dem Zeichen @ abgebildet.

Abweichend von dieser Regel darf in die erste Spalte ein Zeichen $04 (EOT) eingegeben werden, um das Ende des Textes zu markieren (für eine Übertragung via Schnittstelle wichtig).

Betriebsmodi: Der Editor kennt zwei Betriebsmodi, die durch das SD-Kommando des Bedieninterfaces eingestellt werden.

Der Normalmode ist standardmäßig eingestellt. Der CursorESCmode kann mit dem Kommando SD Ax:+1 01 eingestellt werden. Der Normalmode wird mit SD Ax:+1 00 eingestellt.

Die ESC-Sequenzen für Ein/Ausfügungen wurden so gelegt, daß möglichst viele unterschiedliche Terminals benutzt werden können.

Tabelle der vom Editor genutzten Zeichen zur Cursor-steuerung:

Funktion	Normalmode	CursorESCmode
DOWN	$0A oder $16	ESC B
UP	$0B	ESC A
RIGHT	$0C	ESC C
LEFT	$08	ESC D
Insert Char	ESC curs-right	ESC Q
Delete Char	ESC curs-left	ESC W / ESC P
Insert Line	ESC curs-down	ESC E / ESC L
Delete Line	ESC curs-up	ESC R / ESC M

Terminals:	TELEVIDEO TV925 oder kompatible	ITOH CT 101 HP 2648 o.kom.

Funktionen:

Mnemonik	Code	Erklärung
Forward 10 Li	ESC F	Fenster 10 Zeilen weitersetzen
Exit	ESC X	Ausstieg aus dem File, Beendigung des Editors.
Home	ESC H	1. Zeile, 1.Spalte des Files anlaufen und Keybuffer löschen. Der File wird verdichtet.
Ins Keybuffer	ESC I	Der Keybuffer wird vor der Zeile in der der Cursor steht eingesetzt.
Keybuffer ed.	ESC K	Der Keybuffer wird angelaufen und es können in ihn Zeichen eingesetzt werden.
New Numbers	ESC N	Alle Zeilennummern werden aktualisieren und Keybuffer löschen.
Overlay Keyb.	ESC O	Der Keybuffer wird mit der Zeile, in der der Cursor steht, ab der Position des Cursors, gefüllt.
Search Keyb.	ESC S	Es wird in dem File nach dem im Keybuffer enthaltenen Text gesucht. Die Suche beginnt ab der aktuellen Cursorposition in Richtung Fileende.
TabulatorSet	ESC T	Tabulatormarke setzen
Unmask Tabul.	ESC U	Tabulatormarke löschen
Verify Pict.	ESC V	Fenster aus der Datei neu nachladen. Hilft bei unklaren Situation.
Zone Select	ESC Z	Das Fenster wird auf die angegebene Zeilennummer neu positioniert.

Bei den o.a. Escape-Sequenzen dürfen auch kleine Buchstaben mit gleicher Wirkung benutzt werden. Es empfiehlt sich, von Zeit zu Zeit über ESC H eine Verdichtung des Files zu forcieren, insbesondere wenn nur noch wenig Platz im Speicher ist. Bricht der Editor seine Bearbeitung mit der Meldung ..NO MEM. SUSPENDED ab, so kann nach Bereitstellung von genügend Speicher die Task mit CONTINUE fortgesetzt werden. Gelingt dies nicht, so kann der ED-File dennoch gelesen werden, allerdings ist die zuletzt aufgeblätterte Seite nicht im aktuellen Zustand. Es sollte nun unbedingt die Task mit einem UNLOAD entfernt werden, da bei einer Fortsetzung des Editors nach gewaltsamer Veränderung des Files ein Absturz bzw. eine 'BAD POINTER. EXITUS' Meldung erscheint und Veränderungen am File nicht ausgeschlossen sind.

Anmerkungen: Das Bedieninterface ist während einer Editor-Sitzung über CTRL A nicht erreichbar. Sie müssen vorher über ESC X aussteigen. Allerdings kann über die BREAK-Taste dennoch das Bedieninterface bei gleichzeitig aktivem Editor aktiviert werden. Dies ist jedoch nur für den Notfall vorgesehen. Die aktuelle Seite des ED-Files kann über ESC V restauriert werden.

Zum Löschen der Anzeige des Keybuffers muß dieser über ESC K angelaufen werden und dann mit Cursor UP verlassen werden. Der Inhalt bleibt aber weiterhin erhalten (z.B. für die Funktion Suchen: ESC S).

Wenn das verwendete Terminal (z.B. ITOH CT 101) keinen AutoWrapAround-Mode besitzt oder dieser nicht wie oben angegeben funktioniert, so kann mit SD Ax:+1 yy der Editor umparametriert werden. Das Setzen des Bit mit Wertigkeit 2 in yy bewirkt, daß die LF-Generierung nun vom Editor übernommen wird. Das Terminal muß nun so eingestellt werden, daß nach Anschlag des 80.Zeichen in einer Zeile der Cursor in der 80.Spalte stehen bleibt. Es können z.B. mit SD Ax:+1 3; die Funktionen 'curs-ESC' und 'no wrap around' vereinbart werden.

Beispiel: SD A2: 3301; Port2: dialog, LF nach CR, Curs=ESC, IN / OUT

ED ED:marion Der File 'marion' wird editiert.

Nach der Vereinbarung eines 'Working Directorys' (siehe Befehl CD) können die Dateien hierarchisch organisiert werden:

der Nutzer hat das Working Directory ED:ABC angelegt. Die Eingabe von

ED Affe richtet eine Datei mit dem Namen ED:ABC/AFFE ein und diese wird editiert.

Durch die Verwendung von Working Directorys kann im Mehrnutzerbetrieb ein versehentliches Benutzen von Dateien anderer Nutzer vermieden werden. Dazu richten sich alle in einem System arbeitenden Nutzer unterschiedliche Working Directorys ein, über diese erfolgt dann der Zugriff auf die einzelnen den Nutzern zugeordneten Dateien.

Nutzer1 legt mit CD ED:Nutz1 sein Workingdirectory für sich fest, Nutzer2 legt CD ED:Nutz2 fest usw.

Wenn nun beide Nutzer das Kommando:

ED mein eingeben, editieren sie die folgenden im System enthaltenen Files:
Nutzer1: ED:Nutz1/mein und

Nutzer2: ED:Nutz2/mein usw.

Enable Processinterrupt

SYNTAX: ENABLE EV hexnum8;

Beschreibung: Mit dem aus hexnum8 erhaltenen Bitmuster werden die Processinterrupts freigegeben, an deren Bitposition eine '1' in hexnum8 enthalten ist. Es können 32 verschiedene Processinterrupts angesprochen werden.

Beispiele:

```
ENABLE EV 1  Processinterrupt Nr.1 wird freigegeben
ENABLE EV FF Nr.1-16 werden freigegeben
ENABLE EV 0  unsinnig, da ohne Wirkung
```

Hinweis: Diese Anweisung entspricht der gleichnamigen PEARL-Anweisung und erfüllt die gleiche Funktion auf System-Kommandoebene.

SYNTAX: FILES device-list;

Beschreibung: Es werden die Namen der auf den angegebenen Geräten zur Zeit aktiven Files ausgegeben. Bei Floppy /Winchester-Files sind das nur solche, die durch eine ATTACH-Operation (meist implizit) aktiviert wurden. Bei ED-Files werden alle Files des EDFM-Handlers ausgegeben.

Bei Benutzung eines Working-Directorys (siehe CD-Befehl) werden nur die dem vereinbarten Working Directory zugeordneten Files ausgegeben.

Wird ein Gerät angegeben, welches nicht in Files untergliedert ist (siehe SD, DD- Befehl), so wird dies als Kommandofehler behandelt.

Beispiele: FILES ED: FØ: alle aktiven ED: und FØ: Files

FILES F1:

Formatting of a Floppy/Winch.(UH-Fman)

```
SYNTAX:  FORM  S  Floppydevice:forminfo;      (Single density)
         FORM  D  Floppydevice:forminfo;      (double density)
         FORM  D  Winchdev:forminfo;          (Winch nur doub)
```

Beschreibung: Auf dem angegebenen Floppylaufwerk wird neben der eher hardwaretechnischen Softsektorierung auch das Filesystem des UH-Fman installiert. Alle bisher auf der Floppy/Winch. gespeicherten Daten sind unwiderruflich verloren.

'forminfo': Hier wird mitgeteilt, welche Aufteilung in wieviel Sektoren, ob 5" oder 8" und ob ggf. beidseitig formatiert werden soll. Außerdem wird die Anzahl der Tracks mitgeteilt. Insgesamt ergeben sich so 8 Kombinationen für Floppies, wobei 3,5" genau wie 5" behandelt werden:

```
Floppy:      Fx:A8SSyyy -> 26x(128/256 by), 8", singlesided.
-------      Fx:B8SSyyy -> 8x(512/1024 by), 8", singlesided.
             Fx:A8DSyyy -> 26x(128/256 by), 8", doublesided.
             Fx:B8DSyyy -> 8x(512/1024 by), 8", doublesided.
             Fx:A5SSyyy -> 16x(128/256 by), 5", singlesided.
             Fx:B5SSyyy -> 5x(512/1024 by), 5", singlesided.
             Fx:A5DSyyy -> 16x(128/256 by), 5", doublesided.
             Fx:B5DSyyy -> 5x(512/1024 by), 5", doublesided.
Winchester:  Hx:W5Hzyyy -> 256-er Sektor  und z Köpfe/Track.
----------   Hx:X5Hzyyy -> 1024-er Sektor und z Köpfe/Track.

             yyy: 1- bis 3-stellige dezimale Anzahl Tracks.

nur Atari:   Hx:A5H8kkk -> 512-er Sektor siehe  ATARI F-I !!
```

Hinweis: Nach dem Formatieren wird jeder Block (Block=track) gelesen und im Fehlerfall für das Filesystem ausgesondert. Bei mehr als 8 unbrauchbaren Blöcken wird die Operation mit WRONG I/O abgebrochen. Zur Zeit werden die Blöcke aber nicht auf Übereinanderfaltung geprüft (z.B. wenn Drive bei DS nur eine Seite schreiben kann). Bei manchen Laufwerken lassen sich mehr Spuren formatieren als angegeben, Sie gefährden dann aber die Austauschbarkeit zu anderen Systemen.

```
FEHLERMELDUNGEN:  >>...:Fx DRIVE_NOT_READY                 .
                  >>...:Fx DEVICE_WRITE_PROTECTED          .
                  >>...:Fx TRACK_000_NOT FOUND             .
                  >>...:Fx ID_FIELD_NOT_FOUND  (Beim Verify).
```

8" Disketten lassen sich nur auf genügend schnellen Rechnern in double density (16us/byte!) bearbeiten.

Beispiele: FORM S F1:A8SS77; FORM D F0:B5DS80;

Inspect FREE Storage on Disc

SYNTAX: FREE floppy device

Beschreibung: Es wird der noch vorhandene freie Platz auf der angegebenen Floppydisk bzw. Harddisk ausgegeben.

Beispiel: FREE F1:

Hinweis: Wurde ein Working-Directory vereinbart (Siehe Befehl CD), wirkt der Befehl auf das im Directoy vereinbarte Device.

Beispiel: Es wurde das Working-Direcotry F0: vereinbart. Der Befehl FREE wirkt dann auf das Floppydevice F0.

Go and execute

SYNTAX: GO AD hexadr
GO.subtaskname [PRIO integ3] AD hexadr
GO.subtaskname AD hexadr

Beschreibung: Es wird vom System eine Subtask gebildet, die entweder einen Systemnamen 'GOTOxx' oder den angegebenen Tasknamen (2.Form) erhält. Die Task wird mit dem minimal möglichen Workspace ausgestattet und erhält die angegebene Startadresse. Anschliessend wird sie mit der angegebenen Priorität bzw. der Defaultpriorität 50 gestartet.

Beachte: Dies ist eine Hilfskonstruktion für maschinennahe Programmierung. So kann eine kurze Codesequenz schon mal ohne jeden Taskheader zur Ausführung gebracht werden. Auch können im ROM liegende Usertasks ggf. über diesen Weg exekutiert werden. Privilegierte Befehle führen zur Fehlermeldung >> xyz NOT PRIVILEGED.

Beispiele:

```
GO.TEST AD 2000;

GO AD 5000 SZ 5000;

GO.CHECK PRIO 2 AD 8E000;
```

Hinweis: Es lassen sich noch SI und CO - Parameter angeben, die jedoch nur dann Bedeutung haben, wenn der Anwender sich die Informationen aus dem generierten Subtaskheader selbst herausholt.

Durch Angabe eines SZ-Parameters kann die Größe des für die Subtask angeforderten Task-workspace vorbestimmt werden. (siehe Beispiel)

List Task States

SYNATX: L;

Beschreibung: Für jede existierende Task werden folgende Daten aufgelistet:

Adr Prio/User (resident) Status|TWS=xxxx||PC=xxxx| Name

Prio: Gibt die Priorität der Task an. Prioritäten von 0 - 999 sind Anwenderprioritäten. Negative Prioritäten liegen über den Anwenderprioritäten und sind dem Betriebssystem vorbehalten.

User: Gibt an, welchem User die Task zugeordnet ist. User Nummern sind fortlaufend von 0 bis n, entsprechend der Zahl der seriellen I/O-Kanäle vergeben.

Resident: Erscheint bei Tasks, deren Taskworkspace auch nach dem Terminieren der Task erhalten bleibt. Bei zyklisch aktivierten Tasks mit kurzen Einplanungsintervallen kann das Resident Attribut zur Verkürzung der Verwaltungszeiten benutzt werden.

TWS= Hier wird die Adresse des Taskworkspace ausgegeben. Ist sie 00000000, so besitzt die Task noch keinen Workspace. (z.B. hat sie noch keine CPU-Zeit bekommen, oder es war noch zu keinem Zeitpunkt genügend Platz)

PC= Hier wird der letzte vom Dispatcher auf Task-Grundebene gültige und festgestellte Wert des Programmzählers ausgegeben.

Name Gibt den Namen der Task an. Tasknamen, die mit '#' beginnen, sind Systemtasks und lassen sich nicht mit UNLOAD entfernen.

Status: Hier werden die folgenden Abkürzungen eingetragen:

RUN Die Task ist derzeit in der Bearbeitungschlange

CWS? Task wartet auf Zuteilung eines Communication-Elementes, weil sie infolge reger Ausgabetätigkeit auf ein langsameres Gerät ihr Kontingent ausgeschöpft hat.

I/O? Task wartet auf Beendigung einer Ein-/ Ausgabe

SCHD Task ist für eine Aktivierung vorgeplant, die entsprechende Bedingung (Zeit,Ereignis) ist aber noch nicht eingetreten.

SEMA Task wurde durch vergebliches REQUEST auf eine belegt Seamphorvariable blockiert.

SUSP Task wurde suspendiert und wartet auf die CONTINUE-Operation. (Bei System I/O-Tasks wird auf das Ende des laufenden Records gewartet).

DORM Die Task ruht zur Zeit, keine Aktivität

PWS? die Task wartet darauf, daß irgendwo ein passendes Speicherstück frei wird. In der Regel wird auf Procedurworkspace gewartet, es ist aber auch möglich, daß auf ein 'CE' gewartet wird, wobei das zustehende Kontigent (s. CWS? oben) noch nicht ausgeschöft ist.

???? Die Task ist durch mehrere Bedingungen gleichzeitig blockiert, z.B. weil durch die (in PEARL nicht erlaubte) fremdwirkende SUSPEND-Operation die Task zusätzlich verriegelt wurde.

SYNTAX: LOAD
LOAD.subtaskname [PRIO integer3] [loadspeclist]
LOAD [PRIO integer3] [loadspeclist]

Beschreibung: Es wird eine Subtask generiert, deren Name entweder durch 'subtaskname' vorgegeben wird oder mit Namen 'LOADxx' vom System gewählt werden soll. In beiden Fällen kann die Priorität dieser Lader-Task durch eine max. 3-stellige Ganzzahl vorgegeben werden. Bei nicht angegebener Priorität wird ein Wert von 20 eingesetzt.

Fehlt der Zusatz 'loadspeclist', so wird ein Ladevorgang vom Standard-Ladefile des Systems und des Nutzers eingeleitet. Es wird auf den von unten nach oben gesuchten ersten freien passenden Speicherbereich geladen.

Bleiben noch globale Symbole offen, durchsucht der Lader den Scan-Bereich des Systems nach 17-Scheiben, um Referenzen ggf. von dort zu befriedigen. Sind dann immer noch globale Bezüge offen, wird der Lader mit einer entsprechenden Meldung suspendiert.

Der Ladevorgang kann durch die Angabe der 'loadspeclist' parametriert bzw. als bindendes Laden formuliert werden.

loadspeclist ist eine Liste von Geräte/File-Namen und Adreß- oder Größenangaben. Die Elemente dieser Liste werden durch Leerzeichen oder Kommata getrennt.

Adressangabe: AD hexnum8 oder AD=hexnum8

Dabei steht 'hexnum8' für die maximal 8-stellige Hexadezimalzahl, bei der der Lade/Bindevorgang beginnen soll. Erst der hier generierte Lader prüft später, ob der Adreßbereich überhaupt zum Laden verfügbar ist. Diese Adreßangabe ist außer für Testzwecke (glatte Adresse) aus der Sicht von RTOS-UH ein unerwünschter Eingriff.

Programmgröße: SZ hexnum6 oder SZ=hexnum6

Mit 'hexnum6' wird die Größe des ersten zu ladenden Modules vorgegeben und damit die Größenangabe - falls vorhanden - im Ladereingabetext übersteuert. Dies darf natürlich nur so geschehen, daß der SZ-Wert größer als die tatsächliche Modulegröße ist.

Geräte/Filenamen: Es werden die Parameter SI (S-Rekord-Input), SC und LO (List-Output) akzeptiert. Wenn LO nicht angegeben wird, so werden nur fehlende Globalsymbole auf dem Defaultgerät aufgelistet. LO ED:LABEL z.B. bewirkt, daß beim Ladeschluß die Zuordnungstabelle Symbol/Adresse in die Edit-Datei 'LABEL' geschrieben wird.

Mit Hilfe des Parameters SI kann eine Liste von Modulquellen angegeben werden. Die einzelnen Quellen werden durch das Zeichen '+' getrennt. Soweit Querbezüge durch globale Symbole zwischen den einzelnen Modulen existieren, werden sie vom Lader durch einen integrierten Bindevorgang realisiert.

Bsp: SI=ED:LIB1+ED:LIB2 ('SI=' kann entfallen).

Der Parameter SC (Source after Continue) dient zur Angabe einer ggf. benutzten 'Reservedatei', die der Lader immer wieder als Quelle zur Fortsetzung des Bindevorgangs anläuft, solange noch unbefriedigte Globalreferenzen existieren. Nach der Meldung xyz.. SUSPENDED LOADER INPUT wird die Auffüllung dieser Datei bzw. der Defaultdatei (ED:LB bei fehlendem SC) erwartet und anschließend ein CONTINUE-Kommando für die Lader-Subtask. Will man die Referenzen nicht nachreichen, weil man absolut sicher ist, daß sie nicht benutzt werden, so sollte aus Platzgründen die Lader-Subtask mit UNLOAD eliminiert werden.

Beispiele: LOAD.X PRIO 5 AD 6000 SZ 2000 B2:+B2:+F0:QUELLE

Subtaskname ist X, Priorität des Ladevorgangs ist 5. Das erste von Port 2 kommende Modul wird ab Adresse 6000 geladen und auf die Größe von 2000 vergrößert. An dieses Modul wird das nächste von Port 2 stammende Modul angebunden. Die Ablageadresse dieses Moduls ist nicht bekannt. Das gleiche gilt für das dritte von Floppy - Laufwerk 0, File 'QUELLE' - stammende Modul, welches ebenfalls hinsichtlich der globalen Symbole angebunden wird.

LOAD SZ 5000 F1:TEST LO A1: SC B2:

Der File 'TEST' von Floppy Laufwerk 1 wird geladen und die Liste der globalen Symbole samt Adressen über Port 1 ausgegeben. Der Lader hat den Namen LOADxx und läuft mit Priorität 20. Unbefriedigte Globalreferenzen können, falls vorhanden, über das Port 2 ergänzt werden.

Fehlermeldungen: Beim Laden von Dateien, die durch den PEARL-Compiler erzeugt wurden, können bei fehlenden Programmarken oder anderen Fehlern unbefriedigte Vorwärtsbezüge zurückbleiben, die von der Ladertask moniert werden. Daher dürfen nur als fehlerfrei vom Compiler ausgewiesene Module geladen werden!

Doppelt definierte Globalsymbole werden während des Ladevorganges aufgelistet, wobei die Liste nicht vollständig sein muß, weil ein Überlauf des zentralen Meldepuffers möglich ist.

Zeichen, mit denen der Lader nichts anfangen kann, führen zum Abbruch des Ladevorgangs mit der Meldung 'WRONG LOADER INPUT'.

Hinweise: Die Anzahl vorgebbarer Modulquellen ist nur durch die Puffergrenze im Bedieninterface begrenzt.

Mit dem PEARL-Einphasencompiler ohne S=... erzeugte Dateien enthalten in ihrem Kopf nicht die Programmgröße des Modules, sondern einen Pauschalwert (2000). Bei Programmen , in denen nur ein PEARL-Codiertes Modul vorkommt, sollte dieses als erstes geladen werden und die tatsächliche Programmgröße (s. Compilerbilanz) durch 'SZ' vorbesetzt werden.

Es kann eine praktisch unbegrenzte Zahl solcher LOAD-Kommandos abgesetzt werden, die im Multitasking parallel bearbeitet werden. Dabei dürfen diese Ladevorgänge keine gemeinsamen Quellfiles benutzen, da sonst die Eingabedatei wie das Reh von einem Rudel Wölfe 'zerfleischt' würde.

Der von einem solchen Subtasklader gerade bearbeitete Speicherbereich wird während des Ladevorganges als Speichersektion vom Typ 'PWSP' in der Verwaltung von RTOS-UH geführt. Wird eine solche Ladersubtask terminiert (und zweckmäßigerweise auch mit UNLOAD entfernt), so verschwindet auch das zuletzt angefangene Modul aus der Verwaltung.

List User Task

SYNTAX: LU

Beschreibung: Die Anweisung wirkt wie das 'L'-Kommando, jedoch werden die beim Kaltstart des Systems bereits vorhandenen Systemtasks nicht mit aufgelistet.

Beispiel: LU

Make Directory

SYNTAX: MKDIR devicebezeichner:pathlist

Beschreibung: Der Befehl MKDIR erlaubt die Einrichtung von Subdirectorys und ermöglicht damit eine hierarchische Dateiverwaltung. Mit dem Befehl RMDIR lassen sich vereinbarte Directorys wieder löschen.

device-
bezeichner: Es sind Fx: für Disketten- oder Hx: für Harddisklaufwerke erlaubt.

pathlist: 1 bis 7 Buchstaben oder Zahlen pro Element der pathlist. Das erste Zeichen muß ein Buchstabe sein. Die Elemente sind durch '/' zu trennen.

Beispiele: MKDIR F0:USER1

Auf der Diskette in Laufwerk F0: wird das Subdirectory USER1 angelegt. Dort können jetzt Dateien abgelegt werden, z.B. mit

COPY ED:SI>F0:USER1/DATEI1

MKDIR F0:USER1/DATEN1

In dem Subdirectory USER1 wird ein weiteres Subdirectory DATEN1 angelegt.

Hinweise: Mit DIR F0: bekommt man jetzt alle Dateien und Directorys der Root-Ebene. Directorys sind am nachgestellten Schrägstrich zu erkennen. Mit DIR F0:USER1 erhält man die Dateien und weiteren Subdirectorys des Directorys USER1.

No Tracing for specified Task

SYNTAX: NOTRACE taskname,taskname ...

Beschreibung: Die angegebene Task wird aus dem Trace-Mode entlassen (siehe Trace). Dabei wird die Breakpointadresse gelöscht.

Befand sich die angegebene Task nicht im Trace-Mode, so ist die Anweisung ohne Wirkung.

Wie bei der TRACE-Anweisung wird der aktuelle Laufzustand der Task durch diese Anweisung nicht geändert.

Beispiele: NOTRACE TEST XYZ

NOTRACE INIT

Output-device specification

SYNTAX: O device

Beschreibung: Die Ausgabe des Bedieninterfaces wird auf das angegebene Gerät umgeschaltet. Alle Ausgaben des Bedieninterfaces (außer Fehlermeldungen), die durch die folgenden Kommandos dieser Zeile veranlaßt werden, erfolgen auf dem angegebenen Gerät.

device: Es werden alle RTOS-UH bekannten Devices akzeptiert, z.B. A1:, A2:, ED:dir usw.

Beispiele: O A2:;L;S

Es werden die Taskliste und die Speicherbelegung über den Port 2 ausgegeben.

O ED:TEST;DIR F0:;O F1:X;S

Directory von Laufwerk F0: in die Datei ED:TEST schreiben, danach Speicherbelegung auf Floppyfile F1:x.

Compile PEARL- Programm

SYNTAX: PEARL oder P
PEARL.subtaskname [PRIO integer3] [parameterlist]
PEARL [PRIO integer3] [parameterlist]

Beschreibung: Es wird eine flüchtige unabhängige Subtask generiert, deren Name entweder durch 'subtaskname' vorgegeben wird, oder vom System mit 'PCOMxx' bestimmt wird. xx ist eine zweistellige Hexzahl mit automatischer Weiterschaltung. Ebenso kann mit dem Kommando die Priorität der Bearbeitung durch eine 3-stellige Ganzzahl festgelegt werden. Fehlt die PRIO Angabe, wird ein Standardwert von 20 eingesetzt. Fehlt die 'parameterlist', so wird ein Übersetzungslauf gestartet, der von dem Standard-Inputfile des Systems und des Nutzers liest und auf dem entsprechenden Outputfile die S-Records ablegt. Ein Listing wird auf das entsprechende Terminal ausgegeben.

parameterlist ist eine Liste von Geräte/Filenamen und einer Größenangabe. Die Elemente der Liste werden durch Leerzeichen oder Kommata getrennt.

Arbeitsspeichergröße: SZ hexnum6 oder SZ=hexnum6

Mit hexnum6 kann der dynamische Arbeitsspeicher des Compilers bestimmt werden. Der Mindestwert beträgt 500, vom System wird ein Wert von 2800 (entspricht ca. 10 KByte) eingesetzt. Maximal ist ein Wert von 7FFE zulässig. Die gewählte Speichergröße beeinflußt die Übersetzungsgeschwindigkeit praktisch nicht. Faustregel für sehr lange PEARL-Programme (mehr als 4000 Zeilen):

SZ=(Max.Zahl lebender P-Symbole)*14 +2000

Geräte/Filenamen: Es werden die Parameter SI (Source Input), LO (List Output) und CO (Code Output) ausgewertet. Für fehlende Angaben werden die Default-Werte des Systems und Nutzers eingesetzt. Wird der LO-Parameter auf NO gesetzt, so gelten dennoch die für LO gemachten Vereinbarungen, wenn z.B. die Kopf- und Bilanzzeile oder fehlerhafte Programmzeilen ausgegeben werden müssen.

Beispiele: PEARL.X PRIO 40 SZ 5000 F0:TEST>F1:BCOD LO A1:

Das Programm auf Floppylaufwerk 0 in der Datei 'TEST' wird übersetzt. Der Code wird in die Datei 'BCOD' auf Floppylaufwerk 1 geschrieben. Das Übersetzungsprotokoll wird über die Schnittstelle A1: ausgegeben. Name der Subtask ist X.

P

Mit den Defaultwerten für SI (ED:SI für USERØ), LO (A1: für USERØ) und CO (ED:SR für USERØ) wird ein Compiler (Subtaskname PCOMxx) gestartet.

P.PEARL A2:>NO LO A1: LO NO

Der Quelltext wird über Port A2: erwartet, es wird kein Code erzeugt und die Ausgabe fehlerhafter Zeilen erfolgt über Port A1:. Name der Subtask ist PEARL.

Hinweise: Der Compilercode ist wiedereintrittsfest, so daß beliebig viele - sofern Platz für den dynamischen Speicher ist - verschiedene Übersetzungsvorgänge gleichzeitig im Multitasking ablaufen können.

Fehlermeldungen und weitere Eigenschaften des PEARL-Compilers siehe in den Kapiteln C-I ff.

Prevent activation of Task

SYNTAX: PREVENT taskname

Beschreibung: Alle Einplanungen der einzelnen Tasks werden sofort gelöscht. Außerdem werden eventuell bereits im Aktivierungspuffer stehende Neuaktivierungen durch Ausräumen des Puffers verhindert.

Beispiel: PREVENT ABCD,TT

PREVENT init

Hinweis: Trotz gelöschter Zeitplanung bleibt die Aufmerksamkeit der Planungsuhr für den in Aussicht genommenen Zeitpunkt erhalten. Bei 'CLOCK' wird also unter 'NEXT SCHED' eventuell der Termin noch aufgeführt, bleibt aber wirkungslos.

SYNTAX: PROM name;
PROM name*;

Beschreibung: Aus in den RAM-bereich geladenen Programmteilen (Modul oder Task) werden Scheibendaten für das RTOS-UH-Autolink, die Systemkonfigurierung in der Kaltstartphase, erzeugt. Diese Scheibendaten werden als S-Records auf das Ausgabemedium des Bedienинterfaces ausgegeben; für jeden zu 'prommenden' Speicherbereich werden ein S0-Record (enthält Längenangabe), mehrere S2-Records und ein S9-Record erzeugt. S2-Records werden von RTOS-UH stets relativiert, d. h. die Adressangabe im S2-Record zählt stets relativ zur Ladeadresse. Jedem S-Record werden 4 'NUL'-Bytes vorangestellt, um den EPROMmer (meist ohne Protokoll!) nicht zu überrennen. Mit den S-Rekords kann ein EPROM-Gerät (etwa MODIPROG) direkt angesteuert werden, um die Scheibe(n) im Scanbereich im ROM abzulegen (s. E-VII, ggf. Scanbereich erweitern!).
Beim Einsatz des PROM-Befehls sind zwei Einsatzfälle zu unterscheiden:

1) Prommen normal compilierter und geladener PEARL-oder Assembler-Programme
"name" kann ein Modul- oder ein Taskname sein. Folgt einem Modulnamen ein Stern '*' (keine wildcard), so werden alle auf dieses Modul im Speicher unmittelbar folgenden Tasks neben dem Modul ebenfalls bearbeitet.
Es werden Scheibendaten für 13-er Scheiben erzeugt, d. h. es wird ein Speicherbereich quasi als Dump im Eprom abgelegt. Beim Erzeugen des Scheibentextes werden Nulldatenblöcke durch die im 13-Code vorgesehenen Datablöcke weggekürzt. Es empfiehlt sich daher unbedingt, nicht benötigte Initialdaten von Modulvariablen vor dem Absetzen des Befehles zu 'Nullen', etwa durch eine Hilfstask, die man vorher laufen läßt.
Nach Einbau der so erzeugten Scheibe generiert RTOS-UH beim Kaltstart die entsprechenden Module bzw. Tasks an genau die gleichen Adressen, die die Objekte im Moment des PROM-Befehles hatten. Es muß vom Anwender dafür gesorgt werden, daß RAM-Plätze nicht doppelt belegt werden! (RTOS-UH richtet dann den Block, der später vom Scanner erfaßt wird, einfach nicht ein).
Sollen Tasks beim Warmstart automatisch loslaufen, so benutzen Sie vor dem PROM-Befehl den Befehl 'AUTOSTART modulname,taskname'.

2) Prommen von mit der 'CODE=$','VAR=$' - Option erzeugten PEARL-Programmen
"name" kann nur ein Modulname sein; "name*" wird impliziert und muß nicht angegeben werden.
Nach dem Laden des kompilierten Programmes erscheint das Modul incl. aller Tasks als Speicherblock mit der Kennung 'PMDL' beim S-Kommando. Es können keine Tasks aus dem Modul gestartet werden.

Der PROM-Befehl erzeugt nun nur für den Bereich der Modulvariablen eine 13-er Scheibe, die beim Kaltstart des Systems den Modulvariablenblock auf der mit 'VAR=$' angegebenen Adresse einrichtet. Nulldatenblöcke werden, wie im Fall 1), weggekürzt; daher sollte das Programm mit einer Size-Angabe über die 'SC=' Option (Size and Clear) übersetzt werden (spart EPROM-Platz). Tasks werden in eine 1-er Scheibe umgesetzt, Prozeduren ohne Scheibenkennung abgelegt.
Es werden zwei S0- .. S9- Record-Blöcke erzeugt; der erste Block umfaßt die 13-er Modulvariablenscheibe und kann auf beliebiger EPROM-Adresse abgelegt werden, der zweite Block umfaßt den Code-Bereich und muß im EPROM genau beginnend mit der in der 'CODE=$' - Option angegebenen Adresse abgelegt werden. Hierzu kann es ggf. sinnvoll sein, die Ausgabe des PROM-Befehls zunächst in einen ED-File zu lenken, um beide Blöcke mit Hilfe von Editor und COPY zu trennen. Damit wird die Eingabe des Adress-Offset beim EPROMmer erleichtert.
Bei Systemen, die das Betriebssystem aus dem RAM exekutieren (Boot-Systeme wie ATARI oder Transfer aus dem Eprom in das RAM wie bei EMS CPU-4), muß bei der 'CODE=$' - Option die spätere tatsächliche Laufzeit-Adresse angegeben werden. Für EPROM- und RAM- Layout orientiert man sich an den Adress- und Längenangaben der Compiler-Bilanz.
Sollen Tasks beim Warmstart automatisch loslaufen, so benutzen Sie auch hier vor dem PROM-Befehl den Befehl 'AUTOSTART modulname,taskname'.

Die Operation erfolgt auf Ebene der Bedientask, ggf. sollte mit 'O ED:xyz' der S-Rekordtext daher aus Zeitgründen zunächst in eine ED-Datei geschrieben werden. Drücken Sie in der Zeit nicht die BREAK-Taste, die Operation würde sonst abgebrochen!

Beispiele:

```
O A2:;PROM Lager*;O A1:;CLOCK;
```

Modul 'Lager'+folgende Tasks an das Programmiergerät an A2 senden, Uhrzeit erscheint wenn fertig.

```
O ED:Buffer; prom Maus1,Teufel;
```

Task 'Teufel' im Modul 'Maus1' prommen, S-Records in ED-File schreiben.

Print Working Directory

SYNTAX: PWD

Beschreibung: Das mit dem Befehl CD vereinbarte Working Directory kann mit dem Befehl PWD (Print Working Directory) angezeigt werden.

Beispiele: Eingabe : PWD

Antwort : WD= NONE

Es ist kein Working Directory vereinbart.

Eingabe : PWD

Antwort : WD=ED:-

Es ist das Working Directory ED: vereinbart.

Release Semaphorvariable

SYNTAX: RELEASE hexnum8,hexnum8,...

Beschreibung: Durch hexnum8 werden Speicheradressen angegeben, die als Semaphorvariablen in Benutzung sind. Wird dabei der Zustand 'Requested' (and task waiting) verlassen, so wird - falls noch nicht gestorben - die erste (=höchstpriorisierte) wartende Task freigegeben, die auf diese Semaphore wartete. Die Semavariable bleibt in diesem Fall im Zustand 'Requested'. Wird keine wartende Task ermittelt, so wird der Wert der Semavariablen um eins erhöht.

Die Adressen müssen natürlich zunächst ermittelt werden, dazu empfiehlt sich die Benutzung globaler Symbole und Inspektion der Lader-Liste (LO-Option).

Beispiele: RELEASE 4020,10112;

Return Files

SYNTAX: RETURN device-file-list

Beschreibung: Eine Floppy/Winchester-Datei wird aus der Verwaltung von RTOS-UH entlassen. Vorher wird die Datei geschlossen. Die Anweisung wird als Kommandofehler behandelt, wenn ein angegebenes Objekt keine Floppy- oder Winchester-Datei ist.

device-file-list: Es handelt sich um eine Liste, deren Elemente durch Leerzeichen oder Kommata getrennt werden. Ein Element der Liste besteht aus einem Device-Bezeichner (z.B. F0:) und einem Filenamen.

Beispiel: RETURN F1:QUELLE,F0:XYZ

Rewind Files

SYNTAX: REWIND device-file-list

Beschreibung: Die angegebenen Dateien werden zurückgespult. Die Anweisung führt zum Kommandofehler, wenn das Device nicht rückspulbar ist.

device-file-list: Es handelt sich um eine Liste, deren Elemente durch Leerzeichen oder Kommata getrennt werden. Ein Element der Liste besteht aus einem Device-Bezeichner (z.B. F0:) und einem Filenamen.

Beispiel: REWIND F1:QUELLE,F0:XYZ

REWIND ED:myfile

Remove File

SYNTAX: RM device-file-list;

Beschreibung: Die in der Parameterliste angegebenen Dateien werden unwiderruflich aufgegeben. Die Dateien werden aus der Systemverwaltung entfernt, so daß auf sie nicht mehr zugegriffen werden kann. Bei ED-Dateien wird der zur Ablage der Datei benötigte Speicherplatz (Typ 'EDTF') wieder frei verfügbar. In der device-file-list werden vereinbarte Working Directorys mit berücksichtigt.

Beispiele: RM ED:quelle, ED:test, F0:mist;

RM ED:SI;

oder mit Working Directory ED:xyz:

RM mein; löscht ED:xyz/mein

Durch Angabe einer vollständigen Pathlist können Files außerhalb eines vereinbarten Working Directorys gelöscht werden.

vereinbartes Working Directory: ED:xyz

RM ED:nutz1/abc löscht angegebenen File

Hinweis: Das Bedieninterface übergibt die Kommandos an den jeweils zuständigen File-Handler, nachdem überprüft wurde, ob das angegebene Gerät überhaupt in löschbare Files untergliedert ist (siehe SD, DD Befehl). Ist letzteres nicht erfüllt, so wird die Anweisung mitsamt dem Rest der Kommandozeile zurückgewiesen. Wenn das File nicht vorhanden ist, so erfolgt die Meldung durch den File-Handler, ohne das Bedieninterface zu beeinflußen.
Statt des Kommandos RM kann mit gleicher Wirkung auch ERASE eingegeben werden.

Remove Directory

SYNTAX: RMDIR devicebezeichner pathlist

Beschreibung: Die mit dem Befehl MKDIR vereinbarten Directorys können mit dem Befehl RMDIR wieder entfernt werden. Falls das Directory noch ein Directory oder File enthält, erscheint die Fehlermeldung DIRECORY ACTIVE. In diesem Fall sind zunächst die in der Hierarchie weiter unten stehenden Directorys und Files zu entfernen.

device-
bezeichner: Ein Devicebezeichner kann die Form Fx. für Diskettenlaufwerke, bzw Hx. für Harddisklaufwerke haben.

pathlist: Die pathlist kann die Form buchstabenzahlenfolge/buchstabenzahlenfolge... haben. Siehe dazu auch den Befehl MKDIR.

Beispiele: Mit dem Befehl MKDIR wurde das Directory

F0:USER1/PROJEKT1

vereinbart. Es besteht die Möglichkeit mit

RMDIR F0:USER1/PROJEKT1

das Directory PROJEKT1 zu entfernen. Mit dem Befehl

RMDIR F0:USER1

läßt sich auch das Directory USER1 entfernen.

Storage

SYNTAX: S (Form A)
S A (Form B)

Beschreibung: Es wird die Speicherbelegung des gesamten Systems aufgelistet. Damit ist es jederzeit möglich, sich einen Überblick über die Systemauslastung und Belegung der einzelnen Speicherbereiche anzusehen. Bei Form A werden aufeinanderfolgende gleiche ED-Blöcke zusammengefaßt und bei adr2 mit einem '+' gekennzeichnet, um mehr Übersichtlichkeit zu gewährleisten. Die einzelnen Ausgabezeilen haben folgendes Format:

adr1 -> adr2 type |RESIDENT| |taskname| |filename|

adr1/adr2: 8 -stellige hexadezimale Adresse

type:

- MARK Am Anfang und am Ende des von RTOS-UH verwalteten Bereiches steht je eine Speichersektion dieses Typs.

- FREE Diese Speichersektion ist nicht belegt.

- TASK Es handelt sich um den Code-Körper (oder Scheinkörper bei Tasks im ROM) einer Task mit dem nachfolgend angegebenen Namen.

- ATSK Es handelt sich um eine Auto-Start-Task, die sofort beim Systemstart lauffähig ist.

- TWSP Task-Workspace der angegebenen Task.

- CWSP Communication-Element im Besitz der angegebenen Task. Falls vorhanden wird der File-Name des Elementes ausgegeben.

- PWSP Prozedur-Workspace der angegebenen Task.

- MDLE Es handelt sich um den Kopf eines Modules mit dem nachfolgend angegebenen Namen. Dieser Name kann z.B. der bei PEARL-Programmen mögliche Modulname sein.

- EDTF Dieses Segment ist als Editor-Textfile mit dem nachfolgend angegebenen Namen im Speicher abgelegt.

- PMDL Das Segment ist ein PEARL-Modul, das mit der 'CODE=$','VAR=$' - Option des Compilers übersetzt wurde. Das Modul ist nur zur Bearbeitung mit Hilfe des 'PROM'-Befehls geeignet.

- ???? Die Sektion ist nicht identifizierbar. Das Betriebssystem ist durch eine illegale Operation eines Nutzers praktisch unmittelbar vor dem Zusammenbruch.

Set Device-Parameters

SYNTAX: SD device /+ hexadd-expression/ value;

Beschreibung: Die Parametrierung der durch 'device' bezeichneten Datenstation wird durch das (die) Byte(s) in 'value' ersetzt. Bereits in der Warteschlange des Gerätes stehende Ein- oder Ausgaben werden dadurch nicht mehr verändert. Die Wirkung erfolgt nur durch Information gewisser Softwarepakete über den DVDSC-Trap.

'device': Ein dem System bekannter Stationsname. Dabei wird nur eine 'ldn' generiert; A1: läßt sich also z.B. nicht anders als B1: oder C1: parametrieren.

'hexadd-expression' kann wie beim SM-Befehl benutzt werden, um z.B. das erste Byte unverändert zu lassen etc.

'value': Zur Zeit 2 Bytes mit funktionellen Bits.

1.Byte:

- $80 'Rückspulbar', ED,Fx,Hx, (AS, COPY, LOAD)
- $40 Die Station muß vor der ersten Benutzung explizit mit 'open' und nach der letzten mit 'close' angesprochen werden, etwa F0/F1. (Assembler,Compiler,COPY)
- $20 Jedem endenden 'CR' soll ein 'LF' angefügt werden, etwa A1(=B1,C1). (Assembler, Compiler, COPY, PEARL-runtime,Bedieninterface)
- $10 Die Station ist ein dialogfähiges Datenterminal, etwa A1,A2. (COPY,ED)
- $08 Das Echo soll explizit unterdrückt werden, nur bei den Ax, Bx, Cx sinnvoll. (COPY,PEARL-runtime)
- $04 Die Station erlaubt das Löschen bezeichneter Files, nur bei Fx/Hx oder ED sinnvoll. (Bedieninterface)
- $02 Die Station erlaubt die Ausgabe von Daten. (Bedieninterface)
- $01 Die Station erlaubt die Eingabe von Daten. (Bedieninterface)

2. Byte:

- $80 Die Station akzeptiert ein explizites 'DIR' oder 'FILES'-Kommando, nur Fx/Hx und ED. (Bedieninterface)
- $40 Die Station akzeptiert ein explizites 'FORM'-Kommando, nur Fx,Hx. (Bedieninterface)
- $20 CF-Command allowed. (Bedieninterface)
- $10 RMDIR, MKDIR möglich. (Bedieninterface)
- $08 SYNC, SEEK, SAVEP möglich.(Bedien-if, Hyperp)
- $04 zur Zeit unbenutzt.
- $02 Editor (nur ACIA/SCC): Terminal 'No wrap'-typ
- $01 Editor (nur ACIA/SCC): '' 'cursor by esc'

Bsp:

```
SD A2: 0B;  (A2 no echo,input+output)
SD 22; SD xy 22; Trifft alle Files der akt. WD-Ldn!!
SD A1+1 C0; Vorsicht bei vergessenem Doppelpunkt und WD!
```

Syntax: SHARE [PRIO integer3]

Beschreibung: Der Befehl SHARE dient dazu, ein gleichzeitiges Abarbeiten von gleich priorisierten Tasks zu ermöglichen.

Sind z.B. zwei Compiler vom Nutzer gestartet worden, so werden diese normalerweise nacheinander bearbeitet, da sie die gleiche Priorität haben. Damit sie gleichzeitig bearbeitet werden, wird SHARE mit der PRIO [Taskprio - 1] aufgerufen. Nachdem dieses erfolgt ist, wird die Prozessorkapazität auf beide Compiler aufgeteilt.

Ab dem erstmaligen Aufruf bleibt für die angegebene Priorität die Time-Sharing-Funktion aktiv, bis die SHRDxx-Task entladen oder terminiert wird. Auch ein Abort unterbricht das Time-Sharing.

Beispiel:

```
P xxxx > yyyy ; /* läuft mit Prio 20 */;
P zzzz > uuuu ; /* läuft mit Prio 20 */;
SHARE         ; /* jetzt werden beide bearb.*/;
```

Der Aufruf von SHARE richtet eine Subtask SHRDxx ein, die das eigentliche Scheduling ausführt.

Der Aufruf von SHARE muß mit einer um 1 niedrigeren Priorität erfolgen, als der Level, der beeinflußt werden soll, damit die Share-Task sofort aktiv wird. Andernfalls wird sie erst aktiv, wenn auf ihrer Prioritätsstufe freie Prozessorkapazität vorhanden ist.

WICHTIG: Um z.B. die Prioritätsstufe 30 zu beeinflussen muß:

SHARE PRIO 29 eingegeben werden.

Standard Priorität für SHARE ist 20.

Um verschiedene Prioritäts-Level im Time-Sharing Betrieb zu nutzen, können beliebig viele SHARE-Kommandos abgesetz werden.

Show state of specified Task

SYNTAX: SHOW taskname,taskname,...
SH taskname,taskname,... (Kurzform)

Beschreibung: Es wird eine Zustandszeile für jede der angegebenen Tasks ausgegeben. Der Aufbau dieser Zustandszeile entspricht genau denen beim L-Kommando. Wenn einzelne Tasks der Liste nicht vorhanden sind, so erfolgt die Meldung NOT LOADED.

Beispiele: SHOW abcd ASMB12

SH PCOM45

Set Memory

SYNTAX: SM adr value value ... value
SM adr,value,value

Beschreibung: Es kann der Inhalt einer oder mehrerer Speicherzellen verändert werden. Ist value eine n-stellige Hexadezimalzahl, so werden (n+1)/2 Bytes beginnend bei adr abgelegt, wobei der Ablagezeiger für den nächsten Wert value anschließend um diesen Betrag weiterrückt. Die Zugriffe auf die Speicherzellen finden im User-Mode des Prozessors statt!

adr: 1...8 -stellige Hexadezimalzahl, mit der die Adresse der Speicherzelle angegeben wird. Es können auch mehrere Hexadezimalzahlen angegeben werden, die durch ein '+' Zeichen verbunden sein müssen und vom System addiert werden.

value: 1... 8 -stellige Hexadezimalzahl, mit der der neue Inhalt der Speicherzelle angegeben wird.

Beispiele: SM 14EAF 02 AFFE12CD 3 4

Ab der Adresse 14EAF wird 02AFFE12CD0304 abgelegt.

SM 1000 (keine Operation)

SM 6000+10 4

Auf Adresse 6010 wird das Byte 04 abgelegt.

V O R S I C H T !

Diese Anweisung darf nur mit größter Sorgfalt benutzt werden, da das Bedieninterface nicht prüft, ob evtl. lebenswichtige Daten oder Zeiger des Systems zerstört werden. Diese Zerstörungen können zuerst verborgen bleiben und sich erst später bemerkbar machen.

Suspend specified Tasks

SYNTAX: SUSPEND taskname,taskname,...
SU taskname taskname ... (Kurzform)

Beschreibung: Die angegebenen Tasks werden in den Zustand blockiert, wartend auf CONTINUE, gebracht.

Das Bedieninterface prüft, ob die angegebenen Tasks vorhanden sind, ggf. erfolgt die Meldung NOT LOADED. Eine weitere Analyse des aktuellen Taskzustandes findet nicht statt, d.h. die Tasks werden im aktuellen Zustand eingefroren. Wartete die angesprochene Task z.B. auf die Zuteilung einer Semaphore, so hat sie anschließend eine Doppelblockierung. Ihr Zustand wird dann vom L bzw. SHOW Kommando mit ???? gezeigt.

Beispiele: SUSPEND ABCD TEST

SU init

Hinweise: Diese Form der SUSPEND-Anweisung ist nicht Bestandteil der Sprache Basic-PEARL, aber im RTOS-UH-PEARL implementiert.

Synchronize File-System

Der Befehl ist als Shell-extension in den Floppy- und Winchester-filehandlern vorhanden.

SYNTAX: SYNC Floppy-/Winch-devicelist

Beschreibung: Verwaltungsdaten im Speicher und Inhalt auf den bez. Geräten werden aktuell abgeglichen, so daß bei einem Netzausfall die Daten gesichert sind. Die Zustände der Files werden dabei nicht verändert, die Files also auch nicht geschlossen.

Der Befehl darf gegeben werden, wann immer man das für sinnvoll hält. Eventuell passiert überhaupt nichts, etwa wenn man nur von dem Gerät gelesen hat.

BEISPIEL: SYNC H0:,H1:,H2:,F0:;

Hier werden alle ggf. offenen Files auf den Winch. und Floppylaufwerken synchronisiert, d.h. für diesen Zeitpunkt besteht Übereinstimmung zwischen der File-Verwaltung im Speicher und dem Inhalt auf dem Medium.

HINWEIS: Der Befehl eignet sich, um eine Diskette in großer Eile entnehmen zu können. Dabei wird nach dem SYNC die Verwaltungsinformation gelöscht und man kann danach die Floppy herausnehmen.

SYNC F0:;CF F0:FORGET;Clock;

Der anschließende Clock-Befehl dient nur dazu, um den Abschluß der Aktion erkennbar zu machen.

Terminate specified Tasks

SYNTAX: TERMINATE taskname taskname ...
T taskname,taskname... (Kurzform)

Beschreibung: Die angegebenen Tasks werden beendet. Der von ihnen belegte Prozedur-Workspace wird an das System zurückgegeben. Communication-Elemente, die nicht in einer Ausgabeschlange oder in Bearbeitung einer Inputtask sind, werden ebenfalls sofort an das System zurückgegeben. Die automatische Rückgabe aller anderen Communication-Elemente nach Abschluß der Ein- oder Ausgabe wird vorbereitet. Ist die Task nicht vom Typ 'RESIDENT', so wird auch der von ihr belegte Task-Workspace an das System zurückgegeben.

Bezüglich der Semaphore im Besitz der Task wird wie bei der PEARL-Anweisung TERMINATE verfahren.

Das Bedieninterface prüft nicht, ob die angegebene Task existiert oder ihr Laufzustand eine Terminierung ermöglicht, also sie nicht im Zustand 'DORM' ist. Eine Fehlermeldung wird ggf. vom Betriebs-systemkern erzeugt. Meldungen: NOT LOADED (TERMI) oder NOT ACTIVE (TERMI).

Beispiele: TERMINATE TEST INIT

T RUN

Switch Task to Trace Mode

SYNTAX: TRACE taskname adr (Form A)
TRACE taskname L linenr (Form B)
TRACE taskname L linenr,linenr (Form B)

Beschreibung: Die angegebene Task wird in den Hardware-trace-mode überführt (Form A) bzw. die Programmzeilen-überwachung wird eingeschaltet (Form B). Sobald die durch adr (Form A) angegebene Adresse, bzw. eine der max. zwei angebbaren Zeilennummern (Form B) überlaufen wird, wird die Task durch das Betriebssystem suspendiert und die Meldung

taskname:adr BREAKPOINT (Form A)
taskname:L=linenr BREAKPOINT SUSPENDED (Form B)

ausgegeben. Das Erreichen des Haltepunktes löscht diesen nicht. Nach einer CONTINUE-Anweisung kann also der gleiche Haltepunkt erneut angelaufen werden.

Der Befehl ändert nichts am Laufzustand der Task, d.h. auch inaktive oder eingeplante Tasks dürfen an-gesprochen werden.

War die Task bereits im Trace-Mode, so werden alle alten Haltepunkte gelöscht. Eine vollständige Aufhebung der Programmzähler- (Form A) oder Zeilenüberwachung ist nur mit Hilfe des 'NOTRACE'-Kommandos möglich. Auch nach einem Warmstart des Systems sind bei allen Tasks die Trace-Modi gelöscht.

Form A: Die Task läuft im TRACE-Mode wie in 'Zeitlupe' ab. Jeder Maschinenbefehl löst eine Trace-Exception aus, in der der Adreßvergleich erfolgt.

Instruktionen nach TRAPs können nicht als Haltepunkte erkannt werden.

Form B: Das Hochsprachenprogramm muß (wenigstens teilweise) mit der Markierungsoption der UH-Compiler übersetzt worden sein. /*+M*/ bei PEARL, +M bei FTN.

Die Task wird suspendiert, sobald die erste in der angegebenen Zeile (bzw. einer der beiden, falls zwei angegeben waren) beginnende Anweisung erreicht wird. Die Anweisung wird jedoch noch nicht ausgeführt, sondern als erste nach der CONTINUE-Anweisung ausgeführt.

Die Laufgeschwindigkeit der Task wird durch das Zu- oder Abschalten des Line-Trace-Modus praktisch nicht geändert. Allerdings ist das Compilat an sich bereits durch Benutzung der +M -Option um ca. 5 bis 50% verlangsamt.

adr: 1...8 -stellige Hexadezimalzahl, mit der die Adresse des Breakpoints angegeben wird.

linenr: 1...4 -stellige Dezimalzahl, die die Programmzeilennummer des Haltepuktes bezeichnet. Höhere Zeilennummern als 9999 werden modulo 10000 behandelt. Die syntaktische Kontrolle der Zeilennummer ist hexadezimal, so daß unerkannt auch Zeilennummern, die nie vom Compiler generiert werden können, versehentlich eingegeben werden können. Beide Haltepunkte sind völlig gleichwertig. Wird nur einer angegeben, so existiert ein ggf. früher eingegebener zweiter nicht mehr.

Beispiele: TRACE TEST 2346

Die Task TEST wird in den Hardware-Trace überführt und soll bei Adresse 2346 suspendiert werden.

TRACE ABCD L 55 76

Die Hochsprachtask ABCD soll beim Erreichen der Zeilen 55 oder 76 suspendiert werden.

Hinweise: Mit dem Hardware-Tracer können durchaus auch Hochsprach-Codierte Tasks überwacht werden, wenn folgendes beachtet wird:

- Virtuelle Befehle können nicht erkannt werden.
- Der erste Befehl nach einem virtuellen Befehl kann nicht als Haltepunkt dienen.
- Der To-Virtual-Befehl kann benutzt werden.

Werden Adressen oder Zeilennummern in Prozeduren angewählt, so beeinflußt die TRACE-Anweisung natürlich die anderen Tasks nicht, die diesen Haltepunkt überlaufen.

Es ist möglich - bei unabhängig compilierten Modulen -, daß Programmzeilennummern mehrfach vorhanden sind. In diesem Fall - den man möglichst durch Beschränkung auf einen Modul mit +M-codiertem Code vermeiden sollte - ist nach Erreichen des Breakpoint zusätzlich der PC (oder ersatzweise A6) zu inspizieren, um festzustellen, welche der gleichnummerierten Zeilen getroffen wurde.

Mit der +M-Option übersetzte Programme können nur durch Neucompilation ohne diese Option wieder auf maximale Laufgeschwindigkeit und minimalen Code gebracht werden.

Im Gegensatz zu anderen Konstruktionen wird der Code einer Task oder Prozedur durch die Anwendung der TRACE-Anweisung im Speicher nicht verändert.

Trigger specified Interupt

SYNTAX: TRIGGER EV hexnum8

Beschreibung: Sämtliche Interrupts, die durch das mit hexnum8 bezeichnete Bitmuster angegeben werden, passieren die durch ENABLE/DISABLE eingestellte Interruptmaske des Systems. Ist dort keiner der durch hexnum8 ausgewählten Interrupts freigegeben, so ist die Anweisung ohne Wirkung. Das System unterscheidet nicht, ob ein Interrupt durch TRIGGER oder durch die Hardware ausgelöst wurde. Diese Anweisung eignet sich also hervorragend zum Austesten eines interruptgesteuerten Programms.

hexnum8: 1...8 -stellige Hexadezimalzahl, die das 32 Bit Ereignismuster beschreibt. Bei weniger als 8 Stellen werden links Nullen ergänzt.

Beispiele: Vorgeschichte: WHEN EV 3 C TEST; ENABLE EV 7;
WHEN EV 6 XYZ;

TRIGGER EV 1 die Task TEST wird fortgesetzt.

TRIGGER EV 4 die Task XYZ wird aktiviert.

TRIGGER EV 2 TEST wird fortgesetzt, XYZ aktiviert

TRIGGER EV FFFFFFFF alle 'enabled' Interrupts werden gefeuert

Hinweis: Die Anweisung entspricht der gleichnamigen PEARL-Anweisung.

Unload Tasks or Modules

SYNTAX: UNLOAD taskname modulname ... modulname* ...

Beschreibung: Es werden die in der Liste angegebenen Namen in der Systemverwaltung der Tasks und Module gesucht. Existiert ein Name nicht, so wird die Meldung NOT LOADED abgesetzt und das Kommando mit dem nächsten Element fortgesetzt.

taskname: Die Task wird ausgeplant, gepufferte Aktivierungen werden gelöscht und anschließend wird, falls erforderlich, die Task terminiert. Dann verschwindet sie endgültig aus der Verwaltung von RTOS-UH.

modulname: Folgt dem Modulnamen kein '*', so wird nur die dem Modulnamen zugeordnete Speichersektion entfernt. Vorsicht bei PEARL-Modulen!! Dies kann zum Zusammenbruch des Systems führen, falls anschließend noch Tasks auf Modulvariablen oder Datenstationen in dieser Sektion zugreifen.

Folgt dem Modulnamen ein '*', so werden sowohl die Modulsektion als auch sämtliche (!) in unmittelbarer Speicherfolge liegende Tasks entfernt. Dieser Fall ist hauptsächlich für das großzügige Aufräumen im System gedacht und verlangt eine gewisse Vorsicht, da man vorher feststellen muß, an welcher Stelle die erste Nichttasksektion in der Kette steht.

Beispiel: UNLOAD TEST KALA* BAUER

Hinweise: Systemtasks können weder versehentlich noch absichtlich beeinflußt werden. Bereits in der Ausgabeschlange stehende Communication-Elemente einer Task verbleiben dort und werden weiter bearbeitet, allerdings ist der Task-ID gelöscht, so daß als Besitzer (RTOS) beim 'S'-Befehl ausgegeben wird. Hängende Eingaben müssen abgeschlossen sein, damit das CE frei wird.

When event activate or continue given Task

SYNTAX:
```
WHEN EV hexnum8 ACTIVATE taskname [ PRIO integer3 ]
WHEN EV hexnum8 CONTINUE taskname
WHEN EV hexnum8 taskname [ PRIO integer3 ]
WHEN EV hexnum8 C taskname
```

Beschreibung: Bestehende Einplanungen für eine Aktivierung - bei ...ACTIVATE - bzw. zur Fortsetzung - bei ...CONTINUE - werden gelöscht und die angegebene Einplanung für den mit hexnum8 bezeichneten Prozeß-interrupt wird eingetragen. Wird bei ...ACTIVATE keine Priorität angegeben, so wird die taskeigene Priorität eingesetzt. Die aktuelle Priorität einer gerade laufenden Task wird dadurch jedoch nicht verändert, sondern erst, wenn die Einplanung zur Aktivierung führt.

hexnum8: 1...8 -stellige Hexadezimalzahl, die das 32 Bit Ereignismuster beschreibt. Die Aktivierung bzw. Fortsetzung erfolgt, wenn das Bitmuster eines Interruptereignisses (bzw. einer TRIGGER-Anweisung) mindestens ein gesetztes Bit gemeinsam mit dem durch hexnum8 angegebenen Ereignismuster besitzt (d.h. wenn die 'UND'-Verknüpfung nicht 00000000 ergibt. Bei weniger als 8 Stellen werden links Nullen ergänzt.

Beispiele:

```
WHEN EV 2 XYZ PRIO 20

WHEN EV FFFFFFFF CONTINUE ABCD
```

ABCD wird bei einem beliebigen Prozeßinterrupt, sofern er 'enabled' ist, fortgesetzt.

Hier soll erläutert werden, wie man sich Blockkommandos erzeugen kann, die zu vorgegebenen Zeitpunkten oder auf externe Ereignisse hin zur Ausführung kommen.

Im Schnellkurs wurde bereits beschrieben, wie man mit Hilfe entsprechender PEARL-Tasks Bedienkommandos über die sog. XC-Station einplanen kann. Dieses geht allerdings auch ohne den PEARL-Compiler mit Hilfe des COPY-befehles.

Bsp: Sie wollen die Sequenz 'UNLOAD TEST*;P -- LOAD--LAUF' als Blockkommando anlegen und dieses zum Zeitpunkt 16:00:33 zur Ausführung bringen lassen. UNLOAD ist kein Sohnprozeßbefehl, daher:

1. Schritt. Anlegen eines Kommandofiles, z.B. ED:command.
Mit Hilfe des Editors (oder durch COPY) erzeugen wir einen ED-File mit dem Inhalt 'UNLOAD TEST* ....LAUF;'
Der Blockbefehl darf auch aus mehreren Zeilen bestehen, jedoch müssen zu Subtasks (P,AS,COPY,ED,LOAD) die zeitlich nachfolgend mit '--' vereinbarten Operationen stets in der gleichen Zeile stehen!
Wir schließen den Kommandofile mit EOT in Sp.1 ab.

2. Schritt. Generierung einer planungsfähigen Kopiertask.
Wir geben den Befehl COPY.N ED:command CO XC:--;T N;
Nun haben wir eine schlafende Task mit Namen 'N', die beim Erwachen die Befehle im File ed:command in den Fernbedieninterpreter kopiert, dann allerdings verschwindet, falls sie nicht eingeplant ist. Nebenbei: Der String '--' dient zur Unterdrückung der Abschlußmeldung. Das Verschwinden können wir durch eine Pseudoeinplanung 'When EV 0 ACTIVATE N' verhindern, wenn wir nur das Kommando 'N' als neuen Bedienbefehl benutzen wollen.

In unserem Beispiel planen wir 'N' nun wie folgt ein:

AT 16:00:33 ACTIVATE N;

Sehr viel einfacher gelingt die Einplanung eines einzigen Befehles, wenn dieser einer der subtaskgenerierenden (P,AS,LOAD, etc.) ist. Genau wie im obigen Beispiel die COPY-Task kann dieser Sohnprozeß (Subtask) dann unmittelbar eingeplant werden:

Bsp: P.Z ED:file--Unload micmac*--Load--Run; T Z;
AT 14:00:00 ACTIVATE Z;

Die Leistung ist hier ganz ähnlich: Sobald der Compiler fehlerfrei fertig ist, übernimmt er (ja, er höchstpersönlich!) die Ausführung des UNLOAD und erzeugt einen Lader. Der Lader schließlich startet zum Abschluß die Task 'Run'.

Die mit '--' an Sohnprozesse angeschlossenen Folgebefehle werden immer von diesen selbst exekutiert. Evtl. bricht sich also ein Compiler die Ohren, wenn er ein DM mit Bus-Error machen muß.

This page intentionally left blank.

SPRACHUMFANG UH-PEARL

Die Implementierung orientiert sich an DIN 66253, daher werden hier nur Abweichungen und Erweiterungen sowie nicht nach DIN festgelegte Eigenschaften behandelt. Soweit Erweiterungen außerhalb der Norm liegen, so wurden diese durch ein 'CALL'-Konstrukt (Ohne CALL-code!) realisiert, damit UH-PEARL-Programme portabel geschrieben werden können.

BEZEICHNER: Es sind - wie allgemein üblich - Ziffern, Kleinbuchstaben und Großbuchstaben erlaubt. Bezeichner dürfen jedoch nicht mit einer Ziffer beginnen. Zwischen Groß- und Kleinbuchstaben wird semantisch unterschieden. Die Bezeichner dürfen maximal aus 24 Zeichen zusammengesetzt werden.

SCHLÜSSELWORTE: Für Schlüsselworte können nur Großbuchstaben verwendet werden. (siehe Norm).

MODULGRÖSSE: Der einphasige Compiler kann den für den RTOS-Lader erforderlichen Kopfeintrag der Modulgröße nur mit Hilfe des Programmierers schaffen. Dazu wird vor der MODULE-Anweisung ein spezielles Size-statement plaziert:

- 1) S=$6500; MODULE test; ....

- 2) SC=$6500; MODULE test; ....

Die Hexzahl hinter S wird als Kopfeintrag dem Lader übergeben und ermöglicht ihm später die bedarfsgerechte Platzsuche.
In der Form 2) wird der Speicherplatz vor dem Ladevorgang gelöscht (auf $0000 gesetzt). Damit sind alle Variablen mit einem definierten Wert initialisiert (wohlgemerkt: nur direkt nach dem Laden, nicht vor jedem Start des geladenen Programms!). Besonders sinnvoll ist der Einsatz dieser Option bei der Erzeugung ROM-fähigen Codes (s. u.), da der erforderliche ROM-Speicherplatz minimiert wird.

Wird der S-Parameter nicht angegeben, so wird ein Ersatzwert von $2000 eingesetzt.

Der Compiler prüft bei der Modulbilanz die Einhaltung der durch 'S=$..' vorgegebenen Obergrenze. Wird sie überschritten, so wird ein LIMIT-ERROR ausgegeben und das Programm kann später nur mit zusätzlichem SZ-Parameter beim 'LOAD'-Befehl geladen werden.(Sonst: >>LOADxy: WRONG ADDRESS LOADER INPUT.) Die Modulgröße ist praktisch nach oben nicht begrenzt, da der Compiler automatisch auf Langadressierung umschaltet. Allerdings gibt es bei der Verwaltung des Konstantenpools Besonderheiten, siehe dazu C-II PEARL-Hilf/Test, Big Module option.

ROM-CODE:

Der Compiler erlaubt die RTOS-UH-kompatible Erzeugung ROM-fähigen Codes mit Trennung zwischen (Modul-) Variablen- und Code-Bereich. Hierzu müssen zusätzlich zur SIZE-Angabe Code- und Variablen-Adressen angegeben werden:

S=$size,CODE=$epromadresse,VAR=$ramadresse; oder:

SC=$size,CODE=$epromadresse,VAR=$ramadresse;

Die Hexzahl epromadresse gibt die Startadresse des Codes im EPROM an, die Hexzahl hinter VAR die Adresse, mit der beginnend die Modulvariablen im RAM abgelegt werden sollen.
Beim EPROM- und RAM-Layout orientiert man sich an den Code- und Variablen-Längenangaben des Compilers.

Der mit dieser Option erzeugte Code ist nicht im RAM ablauffähig; nach Laden der S-Records können mit Hilfe des 'PROM'-Befehls Eprommer-geeignete S-Records erzeugt werden.

DATENTYPEN:

Typ:	Länge	typ. Konstante/INIT
FIXED(1...15)	2 Bytes	21527 -500 100(15)
FIXED(16...31)	4 Bytes	471(31) 0(31) -2(31)
FLOAT(1...23)	4 Bytes	1.23456E-05 0.245
FLOAT(23...55)	8 Bytes	3.1414567893617(55)
CHAR(1...255)	n Bytes	'Abcdefgh--XYZ' 'a'
BIT(1...16)	2 Bytes	'01000100'B 'AFFE'B4
BIT(17...32)	4 Bytes	'AFFE1234'B4
DUR(ATION)	4 Bytes	2 HRS 5 MIN 0.4 SEC
CLOCK	4 Bytes	13:45:2.004
dur/clock		'Atom' ist 1 msec!
SEMA	2 Bytes	PRESET(2)
STRUCT	? Bytes	- -
REF typ	4 Bytes	INIT(Identifier)

Bei den Verbunddaten (STRUCT) können Komponentennamen frei gewählt werden. Der Übersetzer verdaut jedoch nur max. 512 gleichzeitig 'lebende' Verbundtypen.

Mit Hilfe der LENGTH-Anweisung kann die Defaultlänge der Objekte FIXED, FLOAT, CHAR und BIT eingestellt werden. Ohne 'LENGTH'-Statement gilt:

```
FIXED = FIXED(15)
FLOAT = FLOAT(23)
CHAR  = CHAR(1)
BIT   = BIT(1)
```

Achtung: Das 'LENGTH'-statement wirkt auch auf die FIXED und FLOAT Konstanten ohne nachgestellte Länge!

DIN-ABWEICHUNGEN: (Gegen Basis-PEARL!)

	Objekt	Abweichung	Wann o.k?
(-)	SIGNALE	Nicht implementiert	Prio 3
(-)	ARRAYS	Parameterübergabe nur IDENT	bleibt
(-)	ARRAYS	Keine Total-E/A	Prio 2
(-)	ARRAYS	Keine Slices	Prio 2
(-)	DATION	Keine Stationsfelder, keine Untergliederung (CYCLIC ..), kein DCL	Prio 3
(-)	BIT/CHAR	Keine Slices	Prio 2
(-)	BIT	CAT für BIT nicht implem.	Prio 1
(+)	BIT/CHAR	Beliebige Expr. im Selektor xy.CHAR(expr.)	
(+)	Time-schedule	AT/AFTER ... ALL ... DURING/UNITIL erlaubt.	
(+)	SUSPEND	Für fremde Task erlaubt: SUSPEND xyz.	
(+)	Interrupts	Als Prozedurparameter zugelassen.	
(+)	PROC(EDURE)	Der Selbstaufruf (Rekursion) ist erlaubt.	
(+)	Multi-module	Mehrere Systemteile können gebunden werden.	
(+)	DUR-Objekte	Division DUR/DUR ergibt FLOAT(23) Resultat.	
(+)	NOW	Direkter Zugriff auf aktuelle Uhrzeit: C-III.	
(+)	FIXED-Objekte	Basis-Operator 'REM' aus Full PEARL implem.	
(+)	E/A-Anweisung	Ausdr./Functions mit eigener E/A möglich.	
(+)	REF-Variable	Zeiger aus Full PEARL m.E. implementiert. Siehe dazu Sprachumfang Seite 5!	
(+)	Hilf/Test	An/abschaltbare Hilfsfunktionen gemäß C-II.	
(+)	TYPE	-Definition aus FULL PEARL, im UH-PEARL sogar für Grunddatentypen erlaubt.	
(+)	PROC ..	Verbunddatentyp (STRUCT) ist als Rückgabedatentyp erlaubt!	
(+)	Einbaufunktionen	Für Basisgrafik, Ein-/Ausgabe.	
(+)	ST(dation)	Statusabfrage Datenstation.	
(+)	SEMASET expr.,sema	Dynamische SEMA-Reinitialisier.	

Achtung: Gleiche Verbunddatentypen mit gleichem TYPE deklarieren, bzw. spezifizieren, sonst keine Totalzuweisung möglich! Als Prozedurwert ist daher nur 'geTYPEter' Mode sinnvoll!

Definition von Datenstationen im SYSTEM-Teil:

Allg: XYZ:LD/ldn,drive/.pathlist(TFU=integer,NE,AI=$hexno.)<->;

ldn = Warteschlangennummer im Betriebssystem RTOS-UH.

drive = Laufwerksnummer (Untergliederung) des Gerätes.

pathlist= Alphanum. string mit Separatoren '/', dient ev. nur als Platzhalter für OPEN .. BY IDF(...). Dann lang genug vorsehen!!

TFU=... Größe der Rekords (Comm.el.), TFU=1 kann für Dialoggeräte sinnvoll sein (Terminal), belastet aber das Betriebssystem mit großem Overhead. Wenn TFU fehlt, so wird der Wert von 128 angesetzt.

NE No error-message-flag. Der Programmierer wird mit der Funktion ST die Fehlerüberwachung in eigener Regie durchführen. Fehlt NE, so werden Fehlermeldungen ausgegeben.

AI=$... Zusatzinformation $xxxx f. E/A-Treiber laut besonderer Beschreibung. Fehlt AI=, so wird ein Wert von Null angenommen. Das linke Byte toggelt (exklusiv-oder) die korrespondierenden MODE-Bits des 'CE'.

Die Klammer (TFU .. AI=) darf wie einzelne Parameter fehlen, die Reihenfolge der Parameter muß aber eingehalten werden. Es muß nicht jede Station über LD/../ definiert werden. Am Ende des Compilers befindet sich eine (nach hinten erweiterbare) Liste mit Stationsnamen, zu denen ldn und drive automatisch generiert werden (siehe dazu A-I-10 , A-II-ff).
Es ist auch erlaubt, Usernamen gleich Systemnamen zu setzen, z.B. genügt ..;A1; statt etwa ..;A1:LD/0,0/<->;

Beispiel für die Anwendung von Datenstationen:

```
MODULE XYZ;

SYSTEM;    Winch1:LD/5,5/.USR/GE/test(TFU=400)<->;
           Myfile:F0.dies/ist/nur/ein/platzhalter;....
           Terminal:  A1.Dialog (TFU=1,AI=$3C00)<->;
           Termreset: A1.Reset <->;
PROBLEM;   SPC (Winch1,Myfile) DATION INOUT ALPHIC CONTROL(ALL);
           SPC Terminal  DATION INOUT ALPHIC CONTROL(ALL);
           SPC Termreset DATION INOUT ALPHIC CONTROL(ALL);

TS1:TASK; DCL mist CHAR(20);
     OPEN Myfile BY IDF(textvariable oder konstante);

     GET mist FROM Terminal BY A(20); /* Lese 20 Zeichen */;
     ....                             /* im Sondermodus  */;
     GET FROM Termreset BY SKIP; /* normalmodus einschalten*/
END;
```

Forts.: Datenstationen im SYSTEM-Teil

Mit dem OPEN .. BY IDF (..) wird der Inhalt der Textvariablen oder Textkonstante als aktueller Filename für die Dation (hier Myfile) eingesetzt. Durch diese Anweisung kann eine Dation für verschiedene Files genutzt werden, ohne die Notwendigkeit das PEARL-Programm zu ändern. Es ist zu beachten, daß bei Benutzung von OPEN .. BY IDF (..) im Systemteil ein genügend langer Dummy-Name für die Dation als Platzhalter vorgesehen wird. Mindestens von der Länge der Textvariablen bzw. -konstanten (max 24 Buchstaben).

Bei der im Beispiel angegebene Dation (hier A1.Dialog mit Usernamen Terminal) wird mit TFU=1 auf Einzelzeichenübertragung geschaltet. Mit AI=$3C werden die normalerweise eingesetzten Mode-Bit's des Compilers ($BA) getoggelt, sodaß bei einer Eingabe (GET xxx FROM Terminal;) ein 'CE' mit einem Mode-Byte ($82) entsteht. Es wird auf die Zeichen CR ($0D) , LF ($0A) und EOT ($04) nicht mehr gesondert reagiert, sie werden in die Input-variable eingetragen. Weiterhin ist die Funktion Supress-Command eingeschaltet, sodaß der Kommandointerpreter nicht mehr durch ein ($01 - $03 , Ctrl A-C) angestoßen wird, dieses gilt solange bis entweder ein BREAK oder ein weiteres GET auf die gleiche Dation (hier A1.Reset) mit anderem Usernamen (hier Termreset) und anderem AI ausgeführt wird.

Besonderheiten bei der formatierten Eingabe ('GET') im UH-PEARL.

* Das System blockt die Eingabe in Sätze je nach der 'TFU' des Systemteiles, im Standard jeweils 128 Zeichen. Ein vorzeitiges Satzende erzeugen im Standard: Carr.rtn, Line-feed und EOT. (Mode-Byte = $B8 , mit Wait). Dies kann durch das linke Byte von AI ggf. modifiziert werden. Siehe dazu 'MODE'-byte unter E-III 'Communication-element'. Die 'AI'-option ist mit allergrößter Vorsicht zu benutzen. Mit jedem 'GET' wird an der alten Stelle - im alten Satz! - weitergelesen. Wenn das unerwünscht ist, sollte das Format mit SKIP beginnen.

* Vorzeitige Eingabefeldbegrenzung bei E,F,B,T,D - Formaten.

  Steht in dem vereinbarten Eingabefeld ('width') ein Komma, so beginnt hinter dem Komma das Eingabefeld für das nächste Eingabedatum.

  Bsp. GET i,j,k,l FROM A1 BY SKIP, (4)F(20);

  Als Eingabe würde die Zeile -100,2,33,5, korrekt akzeptiert. Man braucht also nicht 4 mal 20 Zeichen vorzusehen. Tut man es dennoch, so dürfen natürlich keine Kommas vorkommen.

  Auch das Zeichen Carriagereturn beendet ein Eingabefeld.

* Vorzeitige Eingabefeldbegrenzung beim A-Format.

  Sie findet überhaupt nur statt, wenn im Format des GET das automatisch angepaßte A-Format steht (d.h. ohne Klammer mit Parametern). Beim Format A(x) wird in jedem Fall die vorgesehene Anzahl Zeichen gelesen.

  Das Eingabefeld endet beim freien A-Format vorzeitig, wenn das Zeichen Carriagereturn eingelesen wird. Dieses wird in Space ($20) verwandelt, auch der Rest des einzulesenden String wird mit spaces aufgefüllt.

  Bsp.
```
DCL C CHAR(70);
GET C FROM A1 BY SKIP,A;
```

  Wenn nun ABCDEFGHI(Carr.rtn) eingegeben wird, so steht hinterher im String C der Text ABCDEFGHI mit 61 folgenden spaces. Diese Option war nicht unumstritten, sie erleichtert aber die schnelle Kommandoanalyse für Dialoge mit PEARL-programmen.

Besonderheiten bei der formatierten Ausgabe ('PUT') im UH-PEARL.

Es wird in Sätzen mit Länge der TFU transferiert, im MODE-Byte ist nur das Output-bit (Mode-Byte =$40 , ohne Wait) gesetzt. Durch 'AI' kann ggf. ein Warten auf Ausgabeende erzwungen werden. (Niemals bei INOUT-Stationen!).

Besonderheiten bei Feldzugriffen

Im UH-PEARL können ARRAYS gemäß DIN maximal 3-dimensional deklariert werden. Standardmäßig erlaubt UH-PEARL den Zugriff auf 32768 abzüglich der Länge eines Feldelements in Byte Feldelemente über FIXED(15)-Indizes. Wird beim Feldzugriff auf den ersten Index über eine FIXED(31)-Zahl zugegriffen, so kann auf 2 147 483 648 abzüglich der Länge eines Feldelementes Feldelemente zugegriffen werden (sofern der Speicher reicht). Der maximalen Index für jede einzelne Dimension ist 32767.

Besonderheiten bei der Benutzung von Zeigervariablen (REF).

Zunächst eine WARNUNG: Wer mit REF-Variablen arbeitet, begibt sich in einen Raum, in dem unser Compiler keinen Schutz vor Unfug mehr geben kann! Eine Zeigervariable kann wer weiß wohin zeigen - und damit kann wer weiß was im System zerstört werden. Gleichwohl sind aber mit REFs Konstrukte programmierbar, die ihren (gefährlichen) Einsatz rechtfertigen.

Implementierungsumfang:

Im UH-PEARL dürfen REF-Variable auf skalare Objekte der Grunddatentypen sowie auf Verbunde zeigen. (Adressierte Plätze in Arrays sind Skalare!). Felder aus gleichartigen Zeigern sind ebenfalls zugelassen. Nicht implementiert sind Zeiger auf folgende Objekte:

LABEL, PROC, SEMA, DATION, INTERRUPT

Musterbeispiel:

```
TYPE MENSCH STRUCT(/ Name CHAR(10), Alter FIXED,
            Nachbarlinks REF MENSCH, Nachbarrechts REF MENSCH/);
DCL (Maier, Mueller, Schulze) MENSCH; /* so wohnen sie auch */;
DCL  Arbeitszeiger REF MENSCH INIT (Maier);
```

Operationen mit REF-Variablen.

1. Positionieren des Zeigers:

```
Arbeitszeiger=Maier;
Maier.Nachbarlinks=NIL;
Maier.Nachbarrechts=Mueller;
.......................
Schulze.Nachbarrechts=NIL;
```

2. Vergleich von Zeigern:

```
IF Arbeitszeiger IS Maier THEN .....
IF Maier.Nachbarlinks ISNT NIL THEN ..
```

Beachte: Im UH-PEARL muß nur einer der beiden Partner der Operatoren IS und ISNT ein Zeiger sein. Der zweite Partner muß aber eine Adresse liefern können oder das Element 'NIL' sein.

3. Dereferenzierung: Es wird auf das 'gezeigte' Objekt zugegriffen. Siehe dazu die 'Dereferenzierregel' der nächsten Seite. Dabei kann durch Selektion einer Komponente wieder ein REF-Objekt entstehen.

4. Verschiebung: Mit Hilfe der Einbaufunktion REFADD kann ein Zeiger sehr schnell verschoben werden. (Spezialität des UH-PEARL)

Dereferenzierregel:

Zeiger, denen ein Selektor folgt, werden stets dereferenziert. Dabei kann allerdings erneut ein Zeiger entstehen:

```
Arbeitszeiger.Nachbarrechts.Alter = 25;
Arbeitszeiger.Nachbarlinks.Nachbarlinks.Name='Krueger';
Schulze.Nachbarrechts=Arbeitszeiger;     /* Zeigerzuweisung*/;
```

Zeiger in Ausdrücken und bei Prozeduraufrufen werden stets dereferenziert. Aus diesem Grund ist der Operator CONT nur auf der linken Seite einer Zuweisung implementiert. Ein einzelner Zeiger auf der rechten Seite einer Zuweisung wird natürlich dann nicht dereferenziert, wenn links ein Zeiger (ohne CONT davor!) steht.

```
CONT Arbeitszeiger=Mueller;
```

Hier wird das Objekt, auf das Arbeitszeiger zeigt, mit den Daten des Verbundes Mueller besetzt.

Da bei Proc-calls dereferenziert wird, können REF-Variable nicht als solche an Prozeduren übergeben werden. Implizit kann über einen Zeiger als Teil eines Verbundes und Übergabe dieses Verbundes aber dennoch ein Zeiger in eine Prozedur importiert werden.

WICHTIGE TIPS:

* Initialisieren Sie neu deklarierte Zeiger stets mit einer verwertbaren Objektadresse, eventuell einen Dummy dafür anlegen. Nie langlebige Zeiger auf kurzlebige Objekte zeigen lassen!

* Statt der nicht existierenden Funktionen MALLOC ('C') und NEW (Pascal), legen Sie sich ein genügend großes Datenfeld an, wegen der short-indizes ggf. 2-dimensional deklarieren. Beim Schaffen eines neuen Objektes holt man dieses aus dem Vorrat. Auf diese Weise lassen sich vernetzte Datenbanken etc. sogar auf Massenspeichern samt Zeigern retten, wenn dafür gesorgt wird, daß die Basisadresse des raumspendenden Feldes stets gleich ist. (DCL auf Modulebene und mit Festadresse laden).

```
DCL ENORM(500,500) MENSCH;  /* Pufferplatz für MALLOC, NEW*/
DCL GIBHER REF MENSCH;
.......
GIBHER=ENORM(1,1);  /* Auf erstes Objekt setzen*/;
Nochfrei=250000(31);
.......
Nochfrei=Nochfrei-1(31); IF Nochfrei LT 0(31) THEN .....
POINTER=GIBHER;
CALL REFADD(GIBHER,1);  /* Zeiger verschieben  */;
```

Noch edler ist es, wenn man sich eine Kette der unbenutzten Plätze anlegt und so einzelne Plätze wieder frei geben kann.

Sonderfunktionen des UH-Compilers können entweder 'global' bei der Aktivierung der Compiler-Subtask oder 'lokal' durch besonders aufgebaute Kommentarzeilen aktiviert werden. Zu den globalen Sonderfunktionen gehört z.B. der Übersetzungslauf ohne Codegenerierung. Zu den globalen und lokalen Funktionen ist dagegen die An- und Abschaltung des Übersetzungsprotokolles zu rechnen. Rein lokale Funktionen sind dagegen etwa die Markiereroption (s.u)und die Code-protokollierung.

Codegenerierung unterdrücken.

Die Codegenerierung kann nur global unterdrückt werden. Dies geschieht bei der Aktivierung des Compilers durch CO=NO (siehe dazu B-II 'PEARL'). Eine wesentliche Zeitersparnis ist damit normalerweise zwar nicht verbunden, man erspart sich aber die Bereitstellung einer CO-Datei, wenn man zunächst nur an der syntaktischen Prüfung seines Programmes interessiert ist.

Übersetzungsprotokoll ein-/ausschalten.

Es wird (siehe B-II 'PEARL') global durch LO=NO aus- bzw. mit LO=device:file oder fehlende LO-spec. eingeschaltet. Für die lokale Steuerung ist eine Kommentaranweisung

```
/*+L ...beliebiger Text */;  zum Einschalten, bzw.
/*-L ...beliebiger Text */;  zum Abschalten vorgesehen.
```

Unabhängig davon, ob das Übersetzungsprotokoll eingeschaltet ist oder nicht, werden Programmfehler in jedem Fall in das LO-medium ausgegeben. Auch die fehlerhafte(n) Zeilen erscheinen mitsamt dem Fehlerzeiger.

Codeprotokollierung ein-/ausschalten.

Urpsrünglich für die Überprüfung der korrekten Compilerfunktion gedacht, dann aber im System belassen, existiert eine Option zur Auflistung des generierten 68000- bzw. Hyperproccodes. Sie wird mit

```
/*+P  ... beliebiger Text */;     eingeschaltet und mit
/*-P  ... beliebiger Text */;     wieder ausgeschaltet.
```

In das Übersetzungsprotokoll eingebettet erscheint bei dieser Option für jeden generierten Befehl eine Zeile. Dabei wird der auf den Modulanfang relativierte Programmzähler, der Befehlmnemo sowie die Liste der Operanden (unter Verwendung der PEARL-namen!) ausgegeben. Der generierte Code gehört jeweils zur nächsten protokollierten PEARL-Programmzeile, wird also quasi mit dieser abgeschlossen. Die Hyperprocbefehle sind im Teil für Systemprogrammierer weiter hinten erläutert, die 68000-Maschinenbefehle erscheinen mit modifizierten Mnemos, etwa ADDX statt ADD.L (X=xtend 32 bit), können in der Regel aber leicht identifiziert werden. Mit Flügeln versehene Operationen sind

sog. 'Loader-messages', die i.a. nicht interessieren dürften. (AC, ACCU = Register D0 oder D0+D1)

Markierungsoption ein-/ausschalten.

Der UH-Compiler ermöglicht mit Hilfe dieser Option die statische und dynamische Einbettung von PEARL-Zeilennummern in das generierte Maschinenprogramm. Damit wird z.B. der Zeilenstop auf Hochsprachebene (siehe B-II 'TRACE') ermöglicht, aber auch die Ausgabe der letzten exekutierten PEARL-Zeile im Falle von Laufzeitfehlern vorbereitet. Die Kommentare

```
/*+M  ...beliebiger Text */;  dienen zum  Ein- sowie mit
/*-M  ...beliebiger Text */;  zum Abschalten der Option.
```

Zeilen, die nur ein Fragment einer mehrzeiligen Anweisung enthalten, können nicht markiert werden. Der Compiler setzt vor den Code der ersten PEARL-Anweisung jeder Zeile, die im Bereich eingeschalteter Option liegt, einen 4 byte langen Spezialbefehl. Dieser besteht aus einem TRAP oder Hyperprocbefehl mit nachfolgender Zeilennummer, die mit Hex-digits dezimal zu lesen ist. Aufgabe dieses Befehles ist, die Zeilennummer in eine spezielle Zelle der Task zu schreiben und dabei zu prüfen, ob für die Zeile ein Zeilenstop vorliegt. Wenn es der Platz erlaubt, sollte möglichst die komplette zu prüfende Task mit eingeschalteter Option übersetzt werden, da sonst leicht Fehlinterpretationen über den Fehlerort möglich sind. Da die Zeilennummern nur vierstellig abgelegt werden, können Zeilennummern größer als 9999 nur modulo 10000 Berücksichtigung finden.
Die Geschwindigkeitsverluste durch diese Option sind im Allgemeinen nur bei wenigen Prozent zu vermuten.
Die 'M'-option ermöglicht mit Hilfe des 'DL'-Befehles (B-II) jederzeit Schnappschüsse der aktuellen Zeilennummer laufender Tasks.

Feldindex- und characterselector Test einschalten.

Mit Hilfe dieser 'T'-option generiert der Compiler einen modifizierten Code, der die Einhaltung der Zugriffsgrenzen zur Laufzeit überprüft. Bei mehrdimensionalen Feldern wird dabei jedoch nicht jeder einzelne Index sondern nur der effektive eindimensionale Resultatindex überprüft. Damit kann die Zerstörung von Code oder Daten ausserhalb des Feldes bzw. der Zeichenkette mit Sicherheit unterbunden werden.

```
/*+T beliebiger Text .. */;  Testmode einschalten.
/*-T beliebiger Text .. */;  Testmode ausschalten.
```

Bei Verletzung der durch das originäre DCL fixierten Grenzen wird eine Systemmeldung abgesetzt, die entweder die letzte registrierte Zeilennummer oder -falls nicht vorhanden- die Speicherstelle des Zugriffscodes enthält. Das Programm kann mit CONTINUE fortgesetzt werden, der Zugriff erfolgt dann ersatzweise auf das erste Element des Feldes bzw. der Zeichenkette.

Big module - option.

Soll die Größe des Modules den Wert von 32 kB überschreiten oder sind mehr als 4096 Bytes Konstanten zu erwarten, so kann dem Compiler eine andere Verwaltung des Konstantenpooles aufgezwungen werden:

```
/*+B  */;        Nach jedem Task-/Proc-end Pool leeren.
/*-B  */;        Erst beim MODEND den Pool leeren.
```

Standardmäßig ist die 'B'-option abgeschaltet, da dies den kürzesten Code ergibt. Nach Einschalten der Option dürfen alle Tasks/Procs jeweils bis zu 32 kB langen Code und jeweils bis zu 4096 Bytes Konstanten belegen, d.h. die sonst für das Modul gültigen Grenzen gelten nunmehr pro Task/Proc, was kaum noch als ernsthafte Beschraenkung anzusehen ist.

Anmerkung: Die obigen Grenzen gelten nur für den Maschinencode, Variablenspeicher ist hiervon nicht betroffen.

--> WICHTIG:

Die Überschreitung der 4096 Byte Grenze in einem einzelnen Modul (oder bei +B in einer Task/Proc) für den Konstantenpool wird erst vom Lader gemerkt und etwas irreführend als 'WRONG REC-CHECKSUM' gemeldet. Abhilfe: Die vielen Textkonstanten als Modulvariable mit INIT ablegen, dann reicht der Konstantenpool wohl immer aus.

This page intentionally left blank.

Im Folgenden wird ein Auszug der Zeichenketten gegeben, die für den PEARL-Compiler besondere Bedeutung haben. Nicht aufgeführt sind allgemein übliche Zeichen sowie die Systemnamen der Datenstationen.

A,A(shortintegerexpr.)	In Formatliste: Textformat autom., fest.
ABS float-/integerexpr.	Monadischer Operator: Absolutbetrag.
ACOS(floatexpr.)	Arcus Cosinus Einbaufunktion.
ACTIVATE task (PRIO)	Aktivierung einer Task.
AFTER durationexpr..	Einplanung Task nach Zeitdauer.
ALL durationexpr..	Zyklische Einplanung für Taskaktivierung.
ALPHIC	In SPC von Datenstationen: Alphanumeric.
AI=.. Im Systemteil:	Additional Information for Device-driver.
ALT pearl-statement	Alternative im CASE-Wirkungsbereich.
AND	Dyadischer Operator: Logisch 'UND'.
ASIN(floatexpr.)	Arcus sinus Einbaufunktion.
AT clockexpr.....	Einplanung einer Task auf Zeitpunkt.
ATAN(floatexpr.)	Arcus Tangens Einbaufunktion.
B,B1,B3,B4	In Bitkonst. und Formatlisten: Binär, oktal, hex.
BASIC	In SPC: Datenstation vom Grundtyp (A/D-Wandler).
BEGIN	Neuen Programmblock aufmachen.
BIT, BIT(length)	In DCL und SPC Einführung BIT-Variable.
.BIT(shortintegerexpr.)	Hinter BIT-objekten: Einzelbitselekt.
BY shortintegerexpr.	Innerhalb REPEAT: Inkrement festsetzen.
''	Anweisungsanfang: Anfang REPEAT-Schl.
BY formatlist	Anschluß eines Formates für Ein/Ausg.
CALL procedure	Aufruf einer Prozedur, nicht:Funktion.
CASE shortintegerexpr.	Fallverzweigung mit nachfolgenden ALT.
CAT zwischen zwei expr.	Dyadischer Operator: Textkonkatenation.
CHAR, CHAR(Length)	In DCL und SPC Einführung für Textvar.
CHARACTER, CHARACTER(length)	Langform von CHAR.
.CHAR(shortintegerexpr.)	Hinter Textobjekten: Zeichenselektion.
CLOCK	In SPC/DCL: Datentyp Clock festsetzen.
CLOSE dation	Schließen einer Datenstation.
CONT refvariable	Zugriff auf das 'gezeigte' Objekt.
CONTINUE task	Allein oder nach Schedule: Task forts.
CONTROL(ALL)	In SPC: Beim UH-PEARL für alle ALPHIC-stationen.
COS(floatexpr.)	Cosinus Einbaufunktion.
CSHIFT zwischen expr.	Dyadischer Operator: Cyclic shift.
DATION	In SPC und PROC-Def.: Typ Datenstation.
DCL liste, datentyp ..	Definition von Variablen, Datentypen.
DECLARE liste, datentyp	Langform von DCL.
DISABLE interrupt	Prozeßinterrupt freigeben.
DUR	In DCL und SPC: Einführung Zeitdauerobjekt.
DURATION	Langform von DUR.
DURING duration	Innerhalb eines Schedules: Zeitdauer.
E(...)	In Formatliste: Gleitkommaexponetialformat.
ELSE pearlstatement	Innerhalb IF: Alternativzweig.
ENABLE interrupt	Prozeßinterrupt zulassen.
END;	Ende von REPEAT-,Task-,PROC-,BEGIN-Blöcken.
ENTIER floatexpr.	Monadischer Op.: Abrunden auf Integer.

```
ENTRY, ENTRY(..)          In SPC: Prozedur, Funktion spezifizieren.
EQ  zwischen expr.    Dyadischer Operator: Vergleich erg:bitlexpr.
EVERY duration                      Im UH-PEARL wie ALL behandelt.
EXOR  zwischen expr.     Dyadischer Operator: Log. EXCUSIVES ODER.
EXP(floatexpr.)                          Exponentialeinbaufunktion.
F(...)                                 In Formatliste: Festkommaformat.
FIN;                          Ende von IF- und CASE- Wirkungsbereichen.
FIT  zwischen expr.         Dyadischer Op.: Genauigkeiten anpassen.
FIXED, FIXED(15), FIXED(31)   Defaultlänge, kurze, lange Integer.
FLOAT, FLOAT(23), FLOAT(55)      Defaultlänge, kurze, lange Float.
FOR  name ....                             Anfang einer REPEAT-Schleife.
FORMAT(formatliste)              Definition eines 'Remote'-Formates.
FROM dation                   In TAKE, GET und READ: Stationsanschl.
GE  zwischen expr.            Dyadischer Operator: Greater or Equal.
GET liste FROM dation BY format          Alphanumerische Eingabe.
(CALL) GETPIX(...)        Einbaufunktion: Farbe eines Pixels holen.
GLOBAL                    In SPC und DCL: Objekt modulübergreifend.
GOTO  label                 Immer ein Wort! Springe zu Marke label.
GT  zwischen expr.                Dyadischer Operator: Greater Than.
HRS                         In duration-konstanten: Anzahl Stunden.
IDENT                     In Proc-def.: Objekt im Ident-mode überg.
IF bitlexpr. THEN ..                     Bedingte Programmverzweigung.
IN                                   In SPC für DATION: Eingabegerät.
INDUCE signal          !!   Erkennt Compiler, jedoch nicht implem.
INIT(liste)                          In DCL: Anfangsdaten festlegen.
INOUT                              In SPC für DATION: Ein/Ausgabegerät.
INTERRUPT                        In SPC: Interrupt mit Namen versehen.
INV                              In SPC/DCL: Objekt ist unveränderlich.
IRPT                                       Kurzform von INTERRUPT.
IS                        Vergleich von REF/REF oder REF/Objectadr.
ISNT                      Vergleich von REF/REF oder REF/Objectadr.
LE  zwischen expr.                Dyadischer Operator: Less or Equal.
LENGTH datentyp                   Defaultlänge von Daten vereinbaren.
LD/ldn,drive/.filename      Im Systemteil: Logical device-definiton.
LD(floatexpr.)                       Einbaufunktion: Dualer Logarithmus.
LG(floatexpr.)                 Einbaufunktion: Dekadischer Logarithmus.
(CALL) LINE(....)              Einbaufunktion für schnelle Grundgrafik.
LIST                      In Formaten: Automatische Anp. an Datentyp.
LN(floatexpr.)                 Einbaufunktion: Natürlicher Logarithmus.
LT  zwischen expr.                   Dyadischer Operator: Less Than.
MIN                              In Durationkonstanten: Anzahl Minuten.
MODEND;                                        Ende des PEARL-Modules.
MODULE name;                             PEARL-Modul mit Namen eröffnen.
NE                 Im Systemteil: No error-messages on userterminal.
NE  zwischen expr.                   Dyadischer Operator: Not Equal.
NIL                                            Zeigerwert 'Nichts'.
NOT bitexpr.                Monadischer Operator: Alle Bit invertieren.
NOW                                  Clocktype-result: Aktuelle Uhrzeit.
ON signal          !!   Wird vom Compiler erkannt, aber nicht impl.
OPEN dation                                    Datenstation öffnen.
OR  zwischen bitexpr.                Dyadischer Operator: log. 'ODER'.
OUT pearlstatement            In CASE: Fluchtaussprung wenn außerhalb.
PAGE                                   In Formatliste: Seitenvorschub.
PRESET(integerliste)                 In DCL für Semaphore: Anfangswert.
PREVENT task                   Einplanungen und gepuff. Akt. löschen.
PRIO integerconst.               Bei Activate und Taskdef: Priorität.
PROBLEM;                         Ende Systemteil, Anfang Problemteil.
```

```
PROC, PROC(par.liste)          Prozedur oder Funktion definieren.
PUT liste TO dation BY format                Alphanumerische Ausgabe.
READ liste FROM dation;                        Unformatiertes Lesen.
REENT                       Prozedur- Funktionsattribut: Reentrant.
REF                    In DCL/SPC: Deklaration/Spez. eines Zeigers.
(CALL) REFADD(zeiger,shift)  Einb.fkt:Verschiebung eines Zeigers.
RELEASE semavariable                           Semaphore hochzählen.
REM                         Dyadischer Operator: Remainder bilden.
REPEAT                     Begrenzte oder unbegrenzte Wiederholung.
REQUEST semavariable          Semaphore abfragen bzw. erniedrigen.
RESIDENT                      Taskattribut: 'wird eilig aktiviert'.
RESUME                       Nach Schedule: Task später fortsetzen.
RETURN, RETURN(expr.)                Prozedur-, Funktionsrücksprung.
RETURNS(datentyp)           In Funktionsdef./-spezif: Rückgabetyp.
(CALL) REWIND(dation)           Einbaufunktion: File zurückspulen.
ROUND  floatexpr.     Monadischer Operator: Symm. runden auf Int.
(CALL) SAVEP(dation,longfixed)     Einbaufunktion: Save position.
SEC                       In Duration-konstanten: Anzahl Sekunden.
(CALL) SEEK(dation,longfixed)   Einbaufunktion: File positionier.
SEMA                                  In SPC/DCL: Semaphorvariable.
SEMASET integerexpr.,semavariable    Nicht-PEARL: Sema dyn. Init.
SEND liste TO dation;                 Ausgabe für BASIC-Stationen.
(CALL) SETPIX(....)                    Einbaufunktion Grundgrafik.
SHIFT  zwischen expr.              Dyadischer Op.: Shift Bitstring.
SIGN  duration-,float-,fixedexpr.     Monadischer Op.: Signumfkt.
SIGNAL                                In SPC: Signal mit Namen belegen.
SIN(floatexpr.)                                Einbaufunktion: Sinus.
SKIP                         In Formatlisten: Neue Zeile anfangen.
SPC liste datentypen       Spezifikation anderweitig def. Objekte.
SPECIFY                                              Langform von SPC.
SQRT(floatexpr.)                     Einbaufunktion: Quadratwurzel.
ST(dation)         Besondere Einbaufunktion: Status der Datenstat.
STRUCT(/ ... /)            In SPC/DCL: zusammengesetzte Datentype.
SUSPEND;                                   Eigene Task suspendieren.
(CALL) SYNC(dation)           Einbaufunktion: Synchronisiere File.
SYSTEM;                                      Beginn des Systemteiles.
TAKE liste FROM dation;          Eingabe für BASIC-Datenstationen.
TAN(floatexpr.)                                Einbaufunktion: Tangens.
TASK (PRIO ..)                                  Definition einer Task.
TERMINATE, TERMINATE task            Eigene, fremde Task abbrechen.
TFU=            Im Systemteil bei Dations: Transferunit festlegen.
TO  shortintegerexpr.        Als/in REPEAT: Obergrenze festlegen.
TO dation              In Ausgabeanweisungen: Datenstation angeben.
TYPE identif. datentyp       Definition eines eigenen Datentypes.
UNTIL clockexpr.                 In schedules: zeitliche Obergrenze.
UPB   zwischen expr.  Dyadischer Operator: i-te Upperbound Array.
WHEN interrupt                   Taskeinplanung für Prozeßinterrupt.
WHILE bitlexpr. REPEAT     In Laufschleife oder reine WHILE Schl.
WRITE liste TO dation.                         Unformatierte Ausgabe.
X                                     In Formatliste: Leerzeichen.
```

Außer den nicht aufgeführten Operatoren (+,- etc.) gibt es:

```
><   entspricht CAT
<>        ''    CSHIFT
==        ''    EQ
>=        ''    GE
>         ''    GT
<         ''    LT
<=        ''    LE
/=        ''    NE
//    Ganzzahldivision von FIXED-Objekten, Rest verloren.
```

RTOS-UH PEARL - Syntax

Die Syntax-Beschreibung orientiert sich an der Backus-Naur-Form, einem Verfahren zur Darstellung einer kontextfreien Grammatik.
Im Einzelnen bedeuten:

{ }	eine 0- bis n-malige Wiederholung des von den Klammern eingeschlossenen Teiles
[ ]	das optionale Auftreten des von den Klammern eingeschlossenen Teiles
( )	eine Klammerung wie im normalen Gebrauch der Mathematik.
|	trennt Alternativen, von denen genau eine abgearbeitet werden muß.
<u>unterstrichene</u>	<u>**Fettschrift**</u> kennzeichnet terminale PEARL-Symbole,d. h. Symbole, die nicht weiter entwickelt werden können.
Normalschrift	kennzeichnet nicht-terminale Symbole
Kursivschrift	dient nur zur Erläuterung. Kursiv geschriebene Textstellen werden in den Produktionsregeln nicht berücksichtigt.
::=	bezeichnet eine Produktionsregel zur Auswertung nicht-terminaler Symbole

Sinn und Zweck dieser Beschreibungsform ist es, die Überprüfung eines vorliegenden Programmes auf syntaktische Korrektheit zu ermöglichen. Hierzu wird das Startsymbol "PEARL-Programm" schrittweise gemäß den aufgezeichneten Produktionsregeln in eine Folge von (fett und unterstrichen) gedruckten Terminal-(Schluß-) Symbolen entwickelt.
Z. B. ergibt sich als erster Entwicklungsschritt, daß ein PEARL-Programm aus der Folge einer beliebigen Anzahl von Modulen besteht.
Die weitere Entwicklung ermöglicht eine Syntax-Prüfung.
Nicht erfaßt wird hierbei die Semantik der Sprache.

Hier ist nur eine vereinfachte Version des für den Compilerbau verwendeten Regelwerkes verzeichnet. Es ist nicht sichergestellt, daß alle hier möglichen PEARL-Programme auch vom Compiler akzeptiert werden; ebenso können, auch im Zuge der Compilerentwicklung, Konstruktionen akzeptiert werden, die nicht mit diesen Produktionsregeln erzeugt werden können.

Operatoren und I/O-Formate sind in der Anlage ausführlich beschrieben.

==========

Anmerkung: Alle Produktionsregeln, die Zeigervariable betreffen, sind in diesem Regelwerk noch nicht aufgeführt.

```
binär-Ziffer::= 0 | 1

octal-Ziffer::= binär-Ziffer | 2 | 3 | 4 | 5 | 6 | 7

Ziffer::= octal-Ziffer | 8 | 9

hexadezimal-Ziffer::= Ziffer | A | B | C | D | E | F

Buchstabe::= A | B | C | D | E | F | G | H | I | J | K | L | M |
             N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
             a | b | c | d | e | f | g | h | i | j | k | l | m |
             n | o | p | q | r | s | t | u | v | w | x | y | z

Sonderzeichen::= _ | + | - | * | / | ( | ) | : | . | , | ; | = |
                 < | > | ' | [ | ] | # | & | ! | " | § | % | ? |
                 ^ | ~

Zeichen::= Ziffer | Buchstabe | Sonderzeichen

Name::= Buchstabe [ { Buchstabe | Ziffer } ]

Namenliste::= ( Name { , Name } )

Namen::= Name | Namenliste

Ganzzahl::= Ziffer { Ziffer }

PEARL-Programm::= Modul { Modul }

Modul::=  MODULE  [  Modul-Name ] ; [ Systemteil ] [ Problemteil ]
                                                          MODEND ;

Systemteil::= SYSTEM ; { [ Gerätebenennung ; ] }

Gerätebenenung::= { [ neuer_Geräte-Name :  ] }
      [ Geräte-Name [ Geräte-Eigenschaften ] ] [ <- | <-> | -> ]

Geräte-Eigenschaften::= ( Merkmal [ , Merkmal [ , Merkmal ] ] )

Merkmal::= ( TFU = Ganzzahl ) | NE | ( AI = $ hex-Ganzzahl )

Problemteil::= PROBLEM ; { Genauigkeits-/Längenfestlegung }
                         { Typdefinition }
                         { Spezifikation }
                         { modulglobale_Vereinbarungen }
                         { Formatlisten-Vereinbarungen }
                         { Prozedur }
                         { Task }

Genauigkeits-/Längenfestlegung::= LENGTH  ( FIXED | FLOAT |
     CHAR | CHARACTER | BIT ) ( natürliche_Ganzzahl ) ;

Typdefinition::= TYPE Name ( Grund-Datentyp | Verbund )

Grund-Datentyp::= ( ( FIXED | FLOAT | CHAR | CHARACTER | BIT )
                    [ ( natürliche_Ganzzahl ) ] )                |
                  DUR | DURATION | CLOCK

Verbund::= Typ-Name | ( STRUCT ( (/ | [ ) Komponente
                                  { , Komponente } ( /) | ] ) )
```

```
Komponente::= [ Namen ] [ INV ] ( Grund-Datentyp | Verbund )

Spezifikation::= ( SPECIFY | SPC ) Spezifizier-Element
                                   { , Spezifizier-Element } ;

Spezifizier-Element::=  Namen  ( ( ( INTERRUPT |  IRPT  |  Dation-
                                     Beschreibung ) [ GLOBAL ] ) |
                        ( Datenspez. | Prozedurspez. ) [ GLOBAL ] )

Dation-Beschreibung::=  DATION  (  IN | OUT |  INOUT  )
                         ( ( ALPHIC CONTROL ( ALL ) ) | BASIC )

Datenspez.::= ( ( [ leere_Dimensionenliste ] [ INV ]
        ( Grund-Datentyp | Verbund ) ) | TASK | SEMA ) [ RESIDENT ]

leere_Dimensionenliste::= ( [ , ] [ , ] )

Prozedurspez.::= ENTRY [ ( Parameterbeschreibung { , Parameterbe-
                           schreibung } ) ]
                 [ Funktionstypangabe ] [ RESIDENT ] [ REENT ]

Parameterbeschreibung::= ( [ INV ] Grund-Datentyp ) |
                         Dation-Beschreibung IDENT  |
                         [ leere_Dimensionenliste ] [ INV ]
                         ( Grund-Datentyp | Verbund ) IDENT

Funktionstypangabe::= RETURNS ( Grund-Datentyp | Verbund )

modulglobale_Vereinbarungen::= ( DCL | DECLARE ) Moduldaten
                                                 { , Moduldaten } ;

Moduldaten::= Namen ( ( [ INV ] Grund-Datentyp [ RESIDENT ]
                                  [ GLOBAL ] [ Anfangswerte ] ) |
                ( ( Verbund | Matrix ) [ RESIDENT ] [ GLOBAL ] ) |
                ( SEMA [ RESIDENT ] [ GLOBAL ] [ Voreinstellung ]
              ) )

Anfangswerte::= INIT ( Konstantenwert { , Konstantenwert } )

Matrix::= Dimensionenliste ( Grund-Datentyp | Verbund )

Voreinstellung::= PRESET ( positive_Ganzzahl
                           { , positive_Ganzzahl } )

Dimensionenliste::= ( natürliche_Ganzzahl [ , natürliche_Ganzzahl ]
                      [ , natürliche_Ganzzahl ] )

Konstantenwert::= ( [ + | - ] ( Zahlenkonstante |
                                Zeitdauer-Konstante |
                                Uhrzeit-Konstante    ) ) |
                  Zeichenketten-Konstante                |
                  Bitketten-Konstante

Zahlenkonstante::= Ganzzahl | ( [ Ganzzahl ] . Ganzzahl
                                [ E [ + | - ] Ganzzahl ] )
                   [ ( natürliche_Ganzzahl ) ]

Zeitdauer-Konstante::= [ Ganzzahl HRS ] [ Ganzzahl MIN ]
                       [ Ganzzahl . Ganzzahl SEC ]
```

```
Uhrzeit-Konstante::= Ganzzahl : Ganzzahl : Ganzzahl

Zeichenketten-Konstante::= ' { Zeichen } '

Bitketten-Konstante::= ' ( ( binär-Ziffer { binär-Ziffer } '
                              ( B | B1 ) )                    |
                          ( octal-Ziffer { octal-Ziffer } ' B3 ) |
                          ( hexadezimal-Ziffer
                            { hexadezimal-Ziffer } B4 ))

Formatlisten-Vereinbarung::= Formatlisten-Name : FORMAT
                               ( Formatliste ) ;

Formatliste::= ( R ( Formatlisten-Name ) ) |
               ( Multiplikator ( Formatliste ) ) |
               ( [ Multiplikator ] Format )

Multiplikator::= ( natürliche_Ganzzahl )

Format::= siehe Anhang Formate

einf._Ausdruck::= Variablen-Name | ( [ + | - ] Ganzzahl )

Prozedur::= Prozedur-Kopf Blockrumpf END ;

Prozedur-Kopf::= Prozedur-Name : ( PROC | PROCEDURE )
                 [ Parameterliste ] [ Funktionstypangabe ]
                 [ RESIDENT ] [ REENT ] [ GLOBAL ] ;

Parameterliste::= ( Parameterangabe { , Parameterangabe } )

Parameterangabe::= Parameter-Namen
                   ( Wertbeschreibung | Identitätsfestlegung )

Wertbeschreibung::= [ INV ] Grund-Datentyp

Identitätsfestlegung::= ( Dation-Beschreibung |
                          ( [ leere_Dimensionenliste ] [ INV ]
                            ( Grund-Datentyp | Verbund ) ) IDENT

Blockrumpf::= { blocklokale_Vereinbarungen } { Anweisung }

blocklokale_Vereinbarungen::= ( DCL | DECLARE ) Blockdaten
                                                { , Blockdaten } ;

Blockdaten::= Namen ( [ INV ] Grund-Datentyp [ Anfangswerte ]
                    ( Verbund | Matrix )

Anweisung::= [ Marken-Name : ] nichtmarkierte_Anweisung

nichtmarkierte_Anweisung::= Begin-Block              |
                            Zuweisung                |
                            Ablauf-Steueranweisung   |
                            Ein/Ausgabe-Anweisung    |
                            Prozeduraufruf           |
                            Rücksprung-Anweisung     |
                            Echtzeit-Anweisung       |
                            ;
```

```
Begin-Block::= BEGIN Blockrumpf END ;

Zuweisung::= Zugriffsausdruck ( = | := ) Ausdruck ;

Zugriffsausdruck::= Variablen-Name [ Indexliste ]
                    [ { . Komponenten-Name } ]
                    [ . ( BIT | CHAR | CHARACTER ) ( Ausdruck ) ]

Indexliste::= ( Ausdruck [ , Ausdruck [ , Ausdruck ] ] )

Ausdruck::= Primitivausdruck { dyad._Operator Primitivausdruck }

Primitivausdruck::= [ monad._Operator ] ( Konstantenwert    |
                                          Zugriffsausdruck  |
                                          Funktionsaufruf   |
                                          ( Ausdruck )      )
dyad._Operator::= siehe Anhang Operatoren

monad._Operator::= siehe Anhang Operatoren

Funktionsaufruf::= Funktionsprozedur-Name [ Argumentliste ]

Argumentliste::= ( ( Ausdruck | Argument-Name )
                 { , ( Ausdruck | Argument-Name ) } )

Ablauf-Steueranweisung::=
( IF Ausdruck THEN { Anweisung } [ ELSE { Anweisung } ] FIN ; ) |
( CASE Ausdruck ALT Anweisung { ALT Anweisung }
                [ OUT { Anweisung } ]                  FIN ; ) |
( ( ( [ FOR Laufvariablen-Name [ FROM Ausdruck ] [ BY Ausdruck ]
          TO Ausdruck ] ) | [ TO Ausdruck ] ) [ WHILE Ausdruck ]
                                        REPEAT Blockrumpf END ; ) |
( GOTO Sprungmarken-Name ; )

Ein/Ausgabe-Anweisung::= (
( OPEN | CLOSE )                                                  |
( ( WRITE | PUT | SEND ) [ ( Konstantenwert | E/A-Objekt)
                    { , ( Konstantenwert | E/A-Objekt ) } ] TO ) |
( ( READ | GET | TAKE ) [ E/A-Objekt { , E/A-Objekt } ] FROM )   )
Dation-Zugriffsausdruck [ BY Formatliste ] ;

E/A-Objekt::= Zugriffsausdruck | Variablen-Name

Prozeduraufruf::= CALL Prozedur-Name [ Argumentliste ] ;

Rücksprung-Anweisung::= RETURN [ ( Ausdruck ) ] ;

Echtzeit-Anweisung::= Ereignis-Operation | Task-Steueranweisung |
                      Koordinations-Anweisung

Ereignis-Operation::= ( ENABLE | DISABLE ) Interrupt-Zu-
                      griffsausdruck ;

Task-Steueranweisung::= Startanweisung      |
                        Abbruch-Anweisung   |
                        Anhalte-Anweisung   |
                        Fortsetz-Anweisung  |
                        Ausplanung

Startanweisung::= [ Einplanung ] ACTIVATE Task-Name [ Priorität ] ;
```

```
Priorität::= ( PRIO | PRIORITY ) Ganz-Zahlenkonstante

Einplanung::= ( Zeit-Einplanung [ zyklische_Einplanung ] ) |
                                          Ereignis-Einplanung

Zeit-Einplanung::= ( AT Clock-Ausdruck )  |
                   ( AFTER Dur-Ausdruck )

Ereignis-Einplanung::= ( WHEN Interrupt-Name [ ( Ganzzahl ) ] )

zyklische_Einplanung::= ( ALL | EVERY ) Dur-Ausdruck
                        [ ( UNTIL Clock-Ausdruck ) |
                          ( DURING Dur-Ausdruck )    ]

Abbruch-Anweisung::= TERMINATE [ Task-Name ] ;

Anhalte-Anweisung::= ( ( Zeit-Einplanung | Ereignis-Einplanung )
              RESUME ; ) | ( SUSPEND [ Task-Name ] ; )

Fortsetz-Anweisung::= [ Ereignis-Einplanung | Zeit-Einplannng ]
                                          CONTINUE Task-Name ;

Ausplanung::= PREVENT Task-Name ;

Koordinations-Anweisung::= ( REQUEST | RELEASE ) Sema-Name ;

Task::= Task-Kopf Blockrumpf END ;

Task-Kopf::= Task-Name : TASK [ Priorität ]
                                [ RESIDENT ] [ GLOBAL ]  ;
```

Anhang Operatoren

a) Monadische Operatoren

Syntax	Operandentyp	Ergebnistyp	Bedeutung
+ a	FIXED(p) FLOAT(p) DURATION	FIXED(p) FLOAT(p) DURATION	c:= + a
- a	FIXED(p) FLOAT(p)	FIXED(p) FLOAT(p)	c:= - a
NOT a	BIT(k)	BIT(k)	Invertieren aller bits
ABS a	FIXED(p) FLOAT(p) DURATION	FIXED(p) FLOAT(p) DURATION	c:= |a| (Absolutwert)
SIGN a	FIXED(p) FLOAT(p) DURATION	passt sich dem Ausdruck an	c:= +1 für a>0 0 für a=0 -1 für a<0
ROUND a	FLOAT(p)	FIXED(k)	Runden zur nähesten FIXED
TOFIXED a	CHAR(1) BIT(k)	FIXED(p)	gem. ASCII k Bits linksbündig
TOFLOAT a	FIXED(p)	FLOAT(k)	Trivial
TOBIT a	FIXED(k)	BIT(p)	1:1 Transfer
TOCHAR a	FIXED	CHAR(1)	gem. ASCII
ENTIER a	FLOAT(p)	FIXED(k)	c:= näheste Ganzzahl <= a

b) dyadische Operatoren

Syntax	PRI	Typ a	Typ b	Typ c	
a ** b	1	FIXED(15) FLOAT(p)	FIXED(15) FIXED(15)	FIXED(15) FLOAT(p)	Exponentiation
a UPB b	1	FIXED(15)	Matrix	FIXED	Maximalindex der a-ten Dimension v. b
a FIT b	1	FIXED(p) FLOAT(p)	FIXED(q) FLOAT(q)	FIXED(q) FLOAT(q)	a wird der Länge von b angepaßt

Syntax	PRI	Typ a	Typ b	Typ c	
a * b	2	FIXED(p) FIXED(p) FLOAT(p) FLOAT(p) FIXED(p) DURATION	FIXED(q) FLOAT(q) FIXED(q) FLOAT(q) DURATION FIXED(q)	FIXED(r) FLOAT(r) FLOAT(r) FLOAT(r) DURATION DURATION	c:= a * b r:= max(p,q)
a / b	2	FIXED(p) FIXED(p) FLOAT(p) FLOAT(p) DURATION	FIXED(q) FLOAT(q) FIXED(q) FLOAT(q) FIXED	FLOAT(r) FLOAT(r) FLOAT(r) FLOAT(r) DURATION	c:= a / b r:= max(p,q)
a // b	2	FIXED(p)	FIXED(q)	FIXED(r)	c:= ENTIER ABS (a/b) * SIGN (a/b), also Ganzzahl-Division r:= max(p,q)
a >< b a CAT b	2	CHAR(n)	CHAR(m)	CHAR(k)	Verkettung 'a+b' k=n+m
a + b	3	FIXED(p) FIXED(p) FLOAT(p) FLOAT(p) DURATION DURATION CLOCK	FIXED(q) FLOAT(q) FIXED(q) FLOAT(q) DURATION CLOCK DURATION	FIXED(r) FLOAT(r) FLOAT(r) FLOAT(r) DURATION CLOCK CLOCK	c:= a + b r:= max(p,q)
a - b	3	FIXED(P) FIXED(p) FLOAT(p) FLOAT(p) DURATION CLOCK CLOCK	FIXED(q) FLOAT(q) FIXED(q) FLOAT(q) DURATION CLOCK DURATION	FIXED(r) FLOAT(r) FLOAT(r) FLOAT(r) DURATION DURATION CLOCK	c:= a - b r:= max(p,q)
a SHIFT b	3	BIT(k)	FIXED(15)	BIT(k)	a um b Schritte shiften, 0 nachschieben b>0: nach links b<0: rechts
a CSHIFT b a <> b	3	BIT(k)	FIXED(15)	BIT(k)	a um b Schritte zykl. shiften b>0: nach links b<0: rechts

Syntax	PRI	Typ a	Typ b	Typ c	
a LT b a < b	4	FIXED(q) FIXED(q) FLOAT(q) FLOAT(q) CLOCK DURATION	FIXED(r) FLOAT(r) FIXED(r) FLOAT(r) CLOCK DURATION	BIT(1)	c:='0'B (falsch) c:='1'b (wahr) bei FIXED/FLOAT-Kombinationen wird FIXED zu FLOAT gewandelt
a GT b a > b					
a LE b a <= b					Vergleich wird bei Länge max(q,r) durchgeführt
a GE b a >= b					
a EQ b a == b	5	CHAR(q) BIT(q)	CHAR(r) BIT(r)		ggf. wird mit ' ' oder '0'B nach rechts aufgefüllt
a NE b a /= b					
a AND b	6	BIT(k)	BIT(m)	BIT(m)	bei m>k wird mit '0'B nach rechts aufgefüllt
a OR b	7				
a EXOR b					

c) Einbaufunktionen

Das System stellt wichtige, mathematische Elementarfunktionen zur Verfügung.
Diese werden syntaktisch wie PEARL-Funktionsprozeduren behandelt; sie dürfen im Systemteil nicht spezifiziert werden, da der Zugang zu den Routinen sonst verschüttet ist.
Die Argumente müssen vom Typ FLOAT(23) oder FLOAT(55) sein; der Resultattyp entspricht dem Argumenttyp.

Verfügbar sind:

ACOS	Arcus Cosinus
ASIN	Arcus Sinus
ATAN	Arcus Tangens
COS	Cosinus
EXP	Exponentialfunktion zur Basis e
LD	Logarithmus Dualis (Basis 2)
LG	Logarithmus (Basis 10)
LN	Logarithmus naturalis (Basis e)
SIN	Sinus
SQRT	Quadratwurzel
TAN	Tangens

Weiterhin werden die Funktionen RANF und DRANF zur Verfügung gestellt. Diese Funktionen dienen zur Erzeugung von Zufallszahlen; sie müssen im Systemteil spezifiziert werden. Beschreibung s. Ende dieses Kapitels.

Formate dienen zur formatierten Ein-/Ausgabe. Die Argumente der Formate sind, der Syntax folgend, einf._Ausdrücke.
Bei der Formatbeschreibung werden folgende Symbole verwendet:

- s: Vorzeichen. Bei Ausgabe wird '+' durch ' ' ersetzt, bei Eingabe kann '+' entfallen (Default-Wert).
- Z: Ziffer. Z--Z meint eine beliebige, durch Argument und Format bestimmte Anzahl von Ziffern.
- [] kennzeichnet optionale Elemente.

Bei der Eingabe wird ein Eingabefeld unabhängig von der eventuell größeren Eingabefeldbreite durch ',' abgebrochen.

Es sind folgende Formate verfügbar:

F-Format

Das F-Format verarbeitet FIXED- und FLOAT-Zahlen. Syntax:

F ⟨ Feldbreite [⌟ Nachkommastellen [⌟ Skalierung]] ⟩

Ausgabe:
- Ausgabefeldaufbau: ' sZ--Z[.Z--Z]'
- Es werden Feldbreite Zeichen erzeugt
- Die Ausgabezeichenfolge steht rechtsbündig im Ausgabefeld.
- Wird Nachkommastellen nicht angegeben, so entfällt auch der Dezimalpunkt; bei FLOAT wird geROUNDet.
- Der ausgegebene Wert entspricht der auszugebenden Zahl, multipliziert mit 10**'Skalierung'.
- Führende Nullen werden durch ' ' ersetzt (außer vor dem Dezimalpunkt).

Eingabe:
- Eingabefeldaufbau: ' [[s]Z--Z[.[Z--Z]]] '
- Ein leeres Eingabefeld ergibt 0
- Die Eingabezeichenfolge kann beliebig im Eingabefeld stehen.
- Führende ' ' werden nicht ausgewertet.
- Nachkommastellen und Skalierung defaultieren zu 0
- Nachkommastellen wird nicht ausgewertet, sondern von der tatsächlichen Eingabe bestimmt.
- Abgespeichert wird der Eingabewert mit um Skalierung verschobenem Dezimalpunkt.

E-Format:

Das E-Format verarbeitet FIXED- und FLOAT-Zahlen. Syntax:

E ⟨ Feldbreite [⌟ Nachkommastellen [⌟ Signifikanz]] ⟩

Ausgabe:
- Ausgabefeldaufbau: ' sZ--Z.Z--ZEsZZ'
- Es werden Feldbreite Zeichen erzeugt.
- Die Ausgabezeichenfolge steht rechtsbündig im Ausgabefeld.
- Es folgen Nachkommastellen Zeichen auf den Dezimalpunkt.
- Die Mantisse umfaßt Signifikanz Zeichen.

Eingabe:
- Eingabefeldaufbau: ' [[s][Z--Z.Z--Z][E[s][Z[Z]] '
- Die Eingabezeichenfolge kann beliebig im Eingabefeld stehen.
- Nachkommastellen wird nicht ausgewertet.

A-Format

Das A-Format verarbeitet Zeichenketten. Syntax:

A [(Länge)]

Ausgabe:
- wird Länge nicht angegeben, so wird die genaue Länge der auszugebenden Zeichenkette verwendet.
- Es werden Länge Zeichen erzeugt.
- Die auszugebende Zeichenkette beginnt linksbündig im Ausgabefeld.
- ggf. wird nach rechts mit ' ' aufgefüllt.

Eingabe:
- wird Länge nicht angegeben, so wird maximal die Länge der einzulesenden Zeichenkettenvariable verwendet. Die Eingabe von Carriagereturn beendet die Eingabe vorzeitig.
- wird Länge angegeben, so werden genau Länge Zeichen eingelesen.
- Die Zuweisung an die einzulesende Zeichenkette erfolgt linbksbündig; ggf. wird nach rechts mit ' ' aufgefüllt.

B-Format

Das B-Format verarbeitet Bitketten. Syntax:

(B | B3 | B4) [(Länge)]

Die I/O ist in binärer (B), octaler (B3) oder hexadezimaler (B4) Form möglich.

Ausgabe:
- wird Länge nicht angegeben, so wird die genaue Länge der auszugebenden Bitkette verwendet.
- Es werden Länge Zeichen erzeugt.
- Die auszugebende Bitkette wird rechtsbündig im Ausgabefeld einfetragen.
- ggf. wird nach links mit '0' aufgefüllt.

Eingabe:
- wird Länge nicht angegeben, so wird die Länge der einzulesenden Bitkette verwendet.
- Die Zuweisung an die einzulesende Bitkette erfolgt rechtsbündig; ggf. wird mit '0' nach links aufgefüllt.
- Länge > Länge der einzulesenden Bitkette wird nicht unterstützt.

T-Format

Das T-Format bearbeitet Daten des Typs CLOCK. Syntax:

T (Feldbreite [, Dezimalstellen])

Ausgabe:
- Die Ausgabe erfolgt rechtsbündig im Ausgabefeld.
- Dezimalstellen wird nodulo 3 verwendet.

Eingabe:
- Dezimalstellen wird nicht verwendet.
- Die Eingabe kann beliebig im Eingabefeld stehen, Leerzeichen bei den Doppelpunkten sind erlaubt.

D-Format

Das D-Format bearbeitet Daten des Typs DURATION. Syntax:

D (Feldbreite [, Dezimalstellen])

Ausgabe:
- Die Ausgabe erfolgt rechtbündig in das Ausgabefeld.
- Dezimalstellen wird modulo 3 ausgewertet.

Eingabe:
- Dezimalstellen wird nicht verwendet.
- Die Eingabe kann beliebig im Eingabefeld stehen.
- HRS, MIN, SEC müssen mit mindestens einem Leerzeichen von den umgebenden Zahlen getrennt werden.

LIST-Format

Das LIST-Format bearbeitet Daten jeden Typs. Syntax:

LIST

Normalerweise tut dieses Format genau das, was man erwartet. (Zumindest meistens, und bei nicht zu hohen Ansprüchen)

R-Format

Das R-(Remote-)-Format dient zur Abarbeitung vorher vereinbarter Formate. Syntax:

R (*Format*-Name)

Steuerelemente

Die Steuerelemente dienen zur Ausgabesteuerung. Vorhanden sind

X [(Ganzzahl)]

Ausgabe:	Eingabe:
- Leerzeichen	- überliest Zeichen

SKIP [(Ganzzahl)]

Ausgabe:	Eingabe:
- Zeilenvorschübe (CR und LF)	- überliest bis zur nächsten Eingabezeile

PAGE [(Ganzzahl)] gibt Seitenvorschübe aus (FF)

Ausgabe:	Eingabe:
- Seitenvorschub (FF)	- wie SKIP

Im UH-PEARL sind neben den sogenannten 'monadischen' Operatoren der DIN-Beschreibung darüberhinaus die wichtigsten mathematischen Funktionen als 'Einbaufunktionen' realisiert. Einbaufunktionen werden vom Compiler eingebettet, ohne daß zur Laufzeit Zeitverluste zur Parameterübergabe, Speicherplatzbeschaffung etc. auftreten können. Sie sind infolge der automatischen Anpassung an einfache oder doppelte Floatgenauigkeit auch viel bequemer zu benutzen als dies mit Bibliotheksroutinen möglich wäre. Normalerweise ist der Code der Einbaufunktionen als Scheibe (mathx.y) im RTOS-UH vorhanden. Allerdings können auch bei Fehlen der Scheibe über den Linkloader noch die nötigen Anschlüsse hergestellt werden. Aus diesem Grunde sind in der Tabelle auch die internen Systemnamen der Einsprungadressen angegeben.

ACOS(expression)	Arcus Cosinus	#SACOS,	#DACOS
ASIN(expression)	Arcus Sinus	#SASIN,	#DASIN
ATAN(expression)	Arcus Tangens	#SATAN,	#DATAN
COS(expression)	Cosinus	#SCOS,	#DCOS
EXP(expression)	Exp. funktion Basis e	#SEXP,	#DEXP
LD(expression)	Logarithmus dualis	#SLD,	#DLD
LG(expression)	Logarithmus Basis 10	#SLG,	#DLG
LN(expression)	Logarithmus naturalis	#SLN,	#DLN
SIN(expression)	Sinus	#SSIN,	#DSIN
SQRT(expression)	Quadratwurzel	#SSQRT,	#DSQRT
TAN(expression)	Tangens	#STAN,	#DTAN

Alle der obigen Funktionen müssen FLOAT(23) oder FLOAT(55) Argumente haben. Der Resultattyp ist vom Typ des Argumentes. Die Funktionsnamen werden nicht in SPC etc. erklärt, sondern stehen sofort zur Verfügung. Wenn vom Nutzer gleichnamige Arrayvariable oder Funktionen deklariert bzw. spezifiziert werden, so ist der Zugang zur entsprechenden Einbaufunktion verschüttet. Während der Compilation überprüft der Compiler den Mode (FLOAT!) des in jedem Fall zu umklammernden Argumentes. Nicht SIN(1) sondern SIN(1.0)!

Beispiele:

```
X=SIN(a*b);
Z=LN(25.0(55));          Z muß vom Typ  FLOAT(55) sein!
v=ASIN(SIN(y));          Hauptwerte beachten,nicht v=y!
```

Erscheint nach dem Laden des PEARL-Programmes eines oder mehrere der oben rechts angegebenen Internsymbole (daher '#') als undefiniert, so fehlt die Math-Scheibe und sie müssen den Einbaufunktionsfile beim Laden mit einbinden.

Die Einbaufunktionen können Laufzeitfehlermeldungen erzeugen, die durch Überschreitung von Grenzwerten verursacht sind. Dabei erscheint neben dem Tasknamen (der exekutierenden Task) eine namentliche Bezeichnung der Einbaufunktion, sodaß die Ursache gut zu ermitteln ist.

Die Funktion 'ST' zur Statusabfrage von Datenstationen.

Aufruf: I=ST(Dationname) bzw.
CASE ST(Dationname)+1 etc. wie skalares FIXED(15).

Beschreibung: Der UH-PEARL-Compiler generiert einen Zugriff auf eine FIXED(15)-Zahl, die im SYSTEM-Block der angegebenen Datenstation abgelegt ist. Diese Zahl gibt an, ob und wenn welche Besonderheiten beim letzten 'PUT' oder 'GET' aufgetreten sind. Der ST-Wert wird zu Beginn jedes 'PUT' oder 'GET' auf Null gesetzt. Der Wert Null bezeichnet daher den ereignislosen Betriebsfall.

Mit Hilfe der ST-Abfrage ist es also möglich, z.B. syntaktisch falsche Eingaben programmgesteuert zu erkennen und geeignet darauf zu reagieren, und zwar unabhängig davon, ob die E/A-Fehlermeldungen der Station eingeschaltet sind oder nicht.

Die Funktion kann auch das Ergebnis der Einbaufunktionen 'REWIND' 'SEEK', 'SAVEP', 'APPEND' und 'SYNC' überprüfen. (Setzen ST nicht auf Null! Zuweisung ST=Ø über 'PUT(GET) TO(FROM) xy BY LIST').

Bedeutung der ST-Werte:

Ø: Korrekte ereignislose Funktion.
1: END-OF-FILE. Daten konnten nicht gelesen werden.
2: Read-Error. Verschiedene Ursachen, z.B. File nicht exist.
3: Falsche Eingabe-syntax, z.B. Buchstabe bei Zahleneingabe.
4: Exponent (Float) out of range.
5: FIXED-Input Zahlenüberlauf.
6: B-Format Eingabestring zu lang.
7: Sonderfehler des Drivers (z.B. Timeout etc.)
8: Anzahl auszuführender SKIPS negativ.
9: Anzahl auszuführender X oder PAGE negativ.
10: Überlauf öffnender Klammern im Format.
11: Datentyp und Format passen nicht zueinander.
12: Datum in der Liste, aber kein Datenformat im Format.
13: Überzählige schließende Klammer im Format.
14: Dieses Gerät ist für diese Operation nicht geeignet.
15: Pathlist zu lang. Liste gekürzt.

Wenn im PEARL-Programm ein Objekt mit Namen ST definiert wurde, so ist der Zugang zur Einbaufunktion verschüttet. Die Funktion darf daher (wie alle Einbaufunktionen) nicht mit SPC spezifiziert werden!

Bei der Übertragung eines solchen PEARL-Programmes auf andere PEARL-Systeme kann man sich dann eine kompatible Funktion mit Hilfe dort (eventuell) vorhandener Signale (ON-Blöcke) selbst definieren.

```
Bsp:    LBL:   PUT 'Geben Sie bitte x ein' TO TERM BY SKIP,A;
               GET x FROM TERM BY SKIP,F(10);
               IF ST(TERM) NE Ø THEN GOTO LBL; FIN;
```

Bitmapping Basis-grafik 'SETPIX', 'GETPIX' und 'LINE'.

Aufruf:
```
CALL SETPIX(xpos,ypos,colour);
CALL GETPIX(xpos,ypos,colour);
CALL LINE(x1pos,y1pos,x2pos,y2pos,colour);
```

Beschreibung: In Wirklichkeit handelt es sich hier nicht um Prozeduraufrufe (die Syntax mit dem CALL ist nur wegen der PEARL-kompatibilität gewählt worden), sondern um extrem schnelle Ansprünge (Nur JSR/RTS als overhead!) der globalen Systemroutinen #SSETP/#SGETP/#SLINE wobei die FIXED(15)-Parameter in die Register D0 .....D4 geladen werden. Falls in Ihrem System diese Routinen nicht vorhanden sind, können sie leicht als Maschinenprogramme selbst geschrieben und beim Laden angelinkt werden. (Dabei hat man in den Routinen die Register D0-D7 und A0-A3 frei verfügbar). Im Standard ruft LINE intern die Funktion SETPIX auf, so daß ggf. nur SETPIX und GETPIX angepaßt werden müssen.

SETPIX und GETPIX sind zueinander komplementär, d.h. mit GETPIX wird der dritte Parameter (muß variabel sein!) auf den mit SETPIX oder sonstwie erzeugten Wert gebracht.

LINE führt eine Pixel-Interpolation durch, wobei Start- und Endpixel gesetzt und durch Zwischenpixel verbunden werden.

Das Löschen von Punkten und Linien erfolgt durch Schreiben mit Anwahl der Hintergrundfarbe.

Bsp: (Kasten)
```
CALL LINE(i,j,i+10,j,1);
CALL LINE(i,j,i,j+10,1);
CALL LINE(i+10,j,i+10,j+10,1);
CALL LINE(i,j+10,i+10,j+10,1);
```

Wenn alles in Ordnung ist, erscheint eine (wasserdichte) quadratische Kiste mit der linken oberen Ecke auf Position i,j.

Achtung: Bedenken Sie, daß der Ablauf dieser Funktionen je nach Rechner evtl. mit sehr hoher Geschwindigkeit ausgeführt wird und unter Umständen nichts zu sehen ist, wenn das Objekt zu früh wieder gelöscht wird! (Abhilfe: Operation mit dem Vertikal-Prozeßinterrupt und WHEN ... synchronisieren, so daß mindestens ein Vollbild abgewartet wird).

Wie bei allen Einbaufunktionen, so dürfen auch hier die Objekte SETPIX, GETPIX und LINE nirgends weder deklariert noch spezifiziert werden. Anderenfalls sind die Funktionen nicht mehr zugänglich.

Eine negative Farbe bedeutet, daß das Pixel durch Invertieren des Altzustandes erzeugt wird, nach erneutem Aufruf erscheint dort also wieder das alte Bild.

Besondere I/O-operationen im UH-PEARL.

Auch hier handelt es sich nicht um normale Unterprogrammaufrufe, sondern um Sonderfunktionen des UH-Compilers, die aus Kompatibilitätsgründen zu anderen Systemen das CALL-Konstrukt benutzen. ST wird nur im Fehlerfalle geändert, also nicht auf Null gesetzt.

```
CALL REWIND(dationname);............... Zurückspulen eines Files.
CALL SYNC(dationname);.............Speicher-File-Synchronisation.
CALL SEEK(dationname,longfixed);....Aufsuchen des x.Byte im File.
CALL SAVEP(dationname,longfixed);......Position des Files retten.
CALL APPEND(dationname);......... Anhang an den File vorbereiten.
```

REWIND: Der angegebene File wird zurückgespult. War er nicht vorhanden, so wird er neu eingerichtet. Ist das Gerät nicht rückspulbar, so erfolgt Meldung und ST=14.

SYNC: Der angegebene File sowie alle anderen Files dieses Laufwerkes (Drive) wird mit dem Inhalt auf dem Medium so abgeglichen, daß z.B. danach alle Daten im Falle des Netzausfalles gerettet sind. Der Positionszeiger des Files, der Öffnungszustand etc. bleibt unverändert. Möglicherweise erfolgt überhaupt keine Aktion weil Memory und Medium (Floppy/Winch) bereits übereinstimmen. Im Fehlerfalle: Meldung und ST=14.

SEEK: Der File wird auf die Position gebracht, die durch den FIXED(31)-Wert bezeichnet wurde. Dies kann eventuell sogar hinter die aktuelle Schreibschlußmarke geschehen wenn der File von früherer Schreibaktivität noch grösser angelegt ist als er zur Zeit benutzt wird!

SAVEP: Die aktuelle Position (das wievielte Byte des Files das nächste zu schreibende sein wird) wird in der FIXED(31)-Variablen abgelegt und steht dann für einen späteren SEEK zur Verfügung.

Anmerkung: Das Paar SEEK/SAVEP ermöglicht den Aufbau verketteter Daten auf den Massenspeichermedien mit schnellem Zugriff weil immer nur der Block der jeweils betroffenen Stelle wirklich eingelesen wird. Meiden Sie möglichst die Nähe des File-Endes, da manche Filehandler nach dem Einlesen des letzten Bytes einen Auto-close für den File durchführen und damit das Laufzeitsystem so auf die (nach SEEK) falsche Fährte locken!

APPEND: Verlängern des Files. Zur Zeit in keinem Driver implementiert. Führt auf Meldung und ST=14.

Wie bei allen Einbaufunktionen, so dürfen auch hier die scheinbaren Prozedurnamen nicht deklariert oder spezifiziert werden, da der Übersetzer dann die Nutzerobjekte benutzen würde.

```
Bsp:   CALL SAVEP(MYFILE,I);     .... CALL SEEK(MYFILE,I);
```

Besondere I/O Operation

Prozeduren für den Binär-Transfer von Daten auf alle File orientierten Dationen. Die Prozeduren müssen spezifiziert werden:

```
SPC (WRITE,READ) ENTRY GLOBAL;
```

Für den Aufruf muß die folgende Syntax eingehalten werden:

```
CALL WRITE (dationame,..Variablenliste..); .......Schreiben
CALL READ  (dationame,..Variablenliste..); .......Lesen
```

Die in der Variablen-Liste aufgeführten Variablen werden in der Reihenfolge ihres Auftretens auf die angegebene Dation 'binär' transferiert.
Als Dation sind z.Z. nur Floppy bzw. Winchester und Edit-Files möglich, da diese das Binärformat unterstützen. Bei Edit-Files bitte die Dation 'EDB' ansprechen, da sonst das Binär-Format nicht unterstützt wird.

Variablen-Liste:

- ARRAY-Typen werden komplett übertragen (total E/A)
- es sind alle Variablen-Typen aus PEARL erlaubt
- es können verschiedene Typen gemischt werden
- die Länge der Liste ist nicht begrenzt

Die Prozedur setzt am Ende der I/O Operation das Status-Byte der Dation, welches mit der Funktion ST(...) abgefragt werden kann. (siehe Einbaufunktionen -2-)

Beispiel:

```
MODULE ..; SYSTEM;..FLOPPY: F0.MIST <->;..;
PROBLEM;....
  SPC (WRITE,READ) ENTRY GLOBAL;
......
task1: TASK;.....
DCL FELD(100,20) FIXED;DCL ARR(3) FLOAT;DCL R FLOAT(55);
....
CALL WRITE(FLOPPY,FELD,ARR,R);
....
CALL READ(FLOPPY,FELD,ARR,R);
.....
END;
MODEND;
```

Es ist darauf zu achten, daß beim Lesen und Schreiben der Daten die Parameterlisten übereinstimmen, da sonst falsche Werte zugewiesen werden. Bei Zugriff auf die Floppy/Winch-Laufwerke wird eine TFU von 32766 Byte benutzt. Bei einem Zugriff auf die Dation EDB. wird eine TFU von 128 Byte benutzt.

Eingestellte TFU für die einzelnen Dation:

Edit -Files	EDB.xxxxx	128 Byte

-- für PEARL - Programme wird 'EDB' durch LD/1,1/ angesprochen.

Floppy-Files	Fy.xxxxx	32766 Byte
Winch -Files	Hy.xxxxx	32766 Byte

!! Andere Dations werden derzeit nicht unterstützt. !!

--Bitte Beachten--

Die Prozeduren führen kein REWIND durch, daher muß vor der Benutzung die Position im File durch ein REWIND oder SEEK eingestellt werden.

Die Einbaufunktion 'NOW'.

Bei Verwendung des Symboles NOW in Ausdrücken, E/A-Listen etc. wird ein Objekt vom Typ CLOCK erzeugt, welches als Inhalt die aktuelle Uhrzeit erhält. Der Anschluß ist sehr schnell.

Der Bezeichner 'NOW' darf nicht anderweitig deklariert oder irgendwie spezifiziert werden, da der Compiler sonst die Nutzervereinbarung bevorzugt.

```
Bsp:      PUT  NOW TO A1 BY SKIP,LIST;

          Starttime=NOW;
          ....... (Rechnung)
          Elapsedtime=NOW-Starttime;
```

Die Einbaufunktion 'REFADD'.

Um die in PEARL nicht explizit vorgesehene Manipulation von Zeigervariablen zu ermöglichen, wurde die Einbaufunktion REFADD geschaffen. Es handelt sich allerdings keinesfalls um einen wirklichen Prozeduraufruf, die CALL-Konstruktion wurde nur aus Kompatibilitätsgründen gewählt! Der Compiler generiert hier einen geschwindigkeitsoptimierten Maschinencode ohne JSR etc.

```
CALL REFADD(Pointer,Shift);
```

Pointer: Eine REF-Variable beliebigen Typs.
Shift: Ein FIXED(15) oder FIXED(31) Ausdruck, bzw. Konstante.

Auf die Adreßvariable 'Pointer' wird der Wert Shift*Objektgröße aufaddiert. 'Objektgröße' ist die Anzahl Bytes, aus der der Datentyp, auf den 'Pointer' zeigt, besteht.

Wichtiger Hinweis: Wenn 'Shift' eine Konstante ist, so werden die notwendigen Rechnungen schon zur Compilezeit erledigt und lediglich ein einziger ADD.. bzw. SUB.. -Befehl entsteht. Daher für 'Shift' nicht leichtfertig Ausdrücke oder Variablen verwenden!

Mit Hilfe der Einbaufunktion REFADD können auf einem kleinen Umweg auch Felder mit mehr als 32000 Elementen adressiert werden, weil bei Shift auch die lange FIXED-Type erlaubt ist.

Natürlich darf auch 'REFADD' nicht anderweitig deklariert oder irgendwie spezifiziert werden.

Die Funktion 'ASSIGN' zum Ändern der Dation-Namen

Diese Prozedur muß spezifiziert werden:

```
SPC ASSIGN ENTRY(DATION INOUT ALPHIC CONTROL(ALL) IDENT,
                                        CHAR(24)) GLOBAL;
```

Für den Aufruf muß die folgende Syntax eingehalten werden:

```
CALL ASSIGN (Dation,newDation);
```

In PEARL werden den Dations logische Namen zugewiesen, dieses erfolgt im System-Teil mit der Anweisung:

```
logName : PhysName.Filename  <->;
```

Beispiel:

```
Plotter: LD/12,3/.DRUCK ->;
```

Mit der Anweisung OPEN logName BY IDF (newname) kann der Dation ein neuer Filename zugewiesen werden, es ist nicht möglich, die Ein/Ausgabe ohne Änderung des Quelltextes zu einer anderen Dation zu schicken.

Hierzu wurde die Funktion 'ASSIGN' eingeführt, sie erlaubt die Zuweisung einer neuen Ein/Ausgabe-Dation aus einem PEARL-Programm heraus.

Beispiel: (es soll die Ausgabe der Dation 'Plotter' auf ein Floppy-File umgelenkt werden)

```
SYSTEM;
   Plotter: LD/12,3/.PLOTT ->;
PROBLEM;
   SPC ASSIGN ENTRY(DATION INOUT ALPHIC CONTROL(ALL) IDENT,
                                        CHAR(24)) GLOBAL;
....
es muß nun im PEARL-Programm eine der Anweisungen:
....
     GET  newDation FROM TERMINAL BY A,SKIP;
oder
     newDation = 'F0';
....
  CALL ASSIGN (Plotter,newDation);    stehen, und die Ausgabe
                                      kann auf die angegebene
                                      Dation umgelenkt werden.
```

Falls bei 'newDation' der String 'TY' angegeben wird, so wird als Ein/Ausgabe-Dation das eigene Terminal (von dem das Programm gestartet wurde) benutzt. Für 'newDation' sind sämtliche im System implementierten Dationnamen, sowie die Bezeichner 'Lx' (x steht für die LDN der Dation) und das erwähnte 'TY' erlaubt.

```
-- newDation ist vom Typ CHAR(24)
-- Dation    ist vom Typ DATION
```

Die Funktionen RANF und DRANF zur Erzeugung von Zufallszahlen

Die Funktionen müssen im Systemteil spezifiziert werden:

```
SPC RANF ENTRY ( FIXED(31) IDENT, FIXED(31) IDENT)
                                    RETURNS(FLOAT(23)) GLOBAL;

SPC DRANF ENTRY ( FIXED(31) IDENT, FIXED(31) IDENT)
                                    RETURNS(FLOAT(55)) GLOBAL;
```

Nach Deklaration der Argumente mit dem korrekten Typ lautet der Aufruf:

```
DCL ZufallN FLOAT, ( ZufallNM1, ZufallNM2 ) FIXED;
...
ZufallN=RANF( ZufallNM1, ZufallNM2);
```

Die Funktionen berechnen die neue 'Pseudo'-Zufallszahl ZufallN aus den beiden vorangegangenen Zufallszahlen gemäß folgender Formel:

$$X_n = k_1 * X_{n-1} - k_2 * X_{n-2} \quad (\bmod 2^{31}-1)$$

mit $k_1 = 217\ 828\ 199$, $k_2 = 314\ 159\ 269$

Die erzeugten Zahlen liegen gleichverteilt im Intervall [0,1).

Hat einer der beiden Parameter (oder beide Parameter) beim Aufruf den Wert 0, so wird er (oder werden beide) vor Ausführung der Rechnung auf die aktuelle Uhrzeit (in Millisekunden) gesetzt.
Durch gezielte Vorbesetzung der Parameter lassen sich reproduzierbare Folgen von 'Pseudo'-Zufallszahlen erzeugen.

Die Funktion 'TASKST' zum Feststellen eines Taskstatus

Diese Prozedur muß spezifiziert werden:

```
SPC TASKST ENTRY (CHAR(24)) RETURNS (BIT(32)) GLOBAL;
```

Der Aufruf lautet:

```
DCL Stat BIT(32);
...
Stat = TASKST (Taskname);
```

Die Funktion gibt den Wert 'FFFFFFFF'B4 zurück, wenn eine Task mit dem angegebenen Namen dem System unbekannt ist. Ist eine derartige Task im Speicher vorhanden, so hat Stat.BIT(1) folgende Bedeutung:

'0'B : Task ist in irgendwelche Aktivitäten verwickelt
'1'B : Task ist weder eingeplant noch in irgendwelche Aktivitäten verwickelt, jedoch dem System bekannt.

Genauere Information über den Taskstatus sind in den Bits 17...32 enthalten. Diese Informationen sind allerdings nur sehr eingeschränkter Aussagekraft, da in dem Zeitraum zwischen Funktionsaufruf und Auswertung des Funktionswertes durchaus Statusänderungen möglich sind. Es bedeutet (in PEARL-Notation, d.h. vom höchstwertigsten Bit (=Stat.BIT(1)) bis zum niederwertigsten Bit (=Stat.BIT(32)) numeriert:

Bit-Nr.	Aussage
2..16	keine Bedeutung
17	reserviert
18	Waiting for I/O. Task wartet auf Ein-/Ausgabe
19	Waiting for Activation. Sonderstatus: Task wartet auf Aktivierung.
20	Suspend. Task ist suspendiert.
21	Waiting for CE. Task wartet auf ein CE (Kontingent erschöpft oder Speicher voll).
22	Waiting for Workspace. Task wartet auf Zusteilung von Workspace.
23	Waiting for Sema. Task wartet auf Sema (vergeblicher Request).
24	reserviert.
25	Planned for Activation. Für die Task liegt eine Einplanung irgendeiner Art vor.
26	reserviert.
27	Timed Activate. Task ist zeitlich zur Aktivierung eingeplant.
28	Cyclic-Activate. Für die Task liegt eine Einplanung zur zyklischen Aktivierung vor.
29	When...Activate. Task ist mit 'WHEN' zur Aktivierung eingeplant.
30	Timed Continue. Task ist zeitlich zur Fortführung eingeplant.
31	When...Continue. Task ist mit "WHEN" zur Fortführung eingeplant.
32	Idle-Task. Task ist Idle-Task des Systems.

In der PEARL-Welt werden solche Unterprogramme nicht anders behandelt als solche, die mit dem PEARL-Compiler in anderen Modulen übersetzt wurden. Das Objekt ist korrekt zu spezifizieren, wobei der Zusatz 'GLOBAL' erforderlich ist.

Darüberhinaus können auch globale Objekte im Nicht-PEARL-Modul, z.B. FIXED Variablen im Assemblerprogrammodul, durch das Linking des Laders aus der PEARL-Welt adressiert werden.

Hinsichtlich der korrekten Codierung des Assemblerprogrammes siehe detaillierte Beschreibung unter E-IV.

Soweit FTN-Unterprogramme betroffen sind, sind natürlich die Einschränkungen hinsichtlich der dort möglichen Datentypen und Übergabemodalitäten (nur per IDENT etc.) nicht zu umgehen.

BEISPIEL:

```
MODULE ....;  SYSTEM; ....;

PROBLEM;

  SPC Hilf ENTRY(FIXED IDENT) RETURNS(FLOAT) GLOBAL;
  ....
  task1:TASK; ...
  ....
  x=Hilf(i);
  ....
  END;

MODEND;
```

z.B. mit LOAD ED:SR+ED:Hilfe; zu laden, wenn im File ED:Hilfe das übersetzte Assemblerprogramm abgelegt wurde.

This page intentionally left blank.

Obwohl die Meldungen durchweg selbsterklärend sind, soll im Folgenden speziell dem Systemneuling eine Hilfestellung gegeben werden.

Zu unterscheiden sind 'lokal detektierbare' Fehler und 'bilanz-detektierbare' Fehler.

Lokal detektierbare Fehler.

Der Compiler bettet in das Übersetzungsprotokoll einen Fehler-prompt '*' in einer nach der falschen Zeile stehenden Zusatzzeile ein. Der Prompt steht in der unmittelbaren Nähe der Stelle, an der die Abweichung endgültig - auch unter Ausnutzung möglicher anderer Interpretationen - festgestellt werden kann. Auch bei abgeschaltetem Übersetzerprotokoll erscheint die fehlerbehaftete Zeile zusammen mit der Zusatzinformation.

In der Regel gilt eine fehlerhafte Anweisung als insgesamt nicht vorhanden. Dadurch können Folgefehler entstehen.

/SYNTAX/ Es wurde keine PEARL-Produktionsregel gefunden. Das kann z.B. durch ein falsches Zeichen, etwa in einer Konstanten 3.14.2, oder falsch geschriebenes PEARL Schlüsselwort (TUSK statt TASK etc.) verursacht werden. Dabei gelten gewisse Schlüsselworte nur in der richtigen Blockumgebung als bekannt, z.B. wird bei der Sequenz MODULE; PUT xyz TO ... das 'PUT' nicht akzeptiert, weil es nur auf Task/Proc-ebene benutzt werden kann.

/MODE/ Bei der Verknüpfung von Daten wurden Datentypen, die nicht verknüpfbar sind, benutzt. Zum Beispiel wurde der DURATION-Variablen x der Wert der FLOAT-variablen z durch x=z; zugewiesen. Meist steht der Fehlerprompt hier am rechten Ende der Anweisung, weil der Compiler erst bei der Zuweisung die Zulässigkeit nach Auswertung der rechten Seite feststellen kann.

Es ist auch möglich, daß eine Zeigervariable oder ein adresslieferndes Objekt erwartet wird und vom Compiler nun nicht vorgefunden wird, z.B. bei der Operation 'CONT', 'IS' etc.

/UNDEFINED/ Die Variable bzw. der Prozedurname etc. wurden dem Compiler nicht durch DECLARE oder SPECIFY bekannt gemacht.

/DOUBLE-DEF/ Der Identifier ist bereits im Gebrauch.

/LIMIT/ Der Wert einer Konstanten liegt außerhalb der zulässigen Grenzen, z.B X=45196(15) etc.

/SUBSCRIPTS/ Beim Zugriff auf ein Feld stimmt die Anzahl der Indizes nicht mit denen der Felddefinition-Spezifikation überein, z.B. DCL A(2,3) FIXED; A(I,J,K)=5.

/DATAWAY/ Die Datenstation ist per Definition/Spezifikation oder auf Grund von Kenntnissen des Compilers nicht in der Lage, wie im Text beabsichtigt zu funktionieren. Beisp: SPC xyz DATION OUT ....;..; GET ... FROM xyz.

/TOWHAT?/ Es wurden im INIT mehr Objekte gefunden als in der zugehörigen Deklarationsliste. Der Compiler kann die überzähligen Elemente nicht zuordnen, sie werden ignoriert.

/INVARIANT/ Einem Objekt mit INV-Attribut soll zur Laufzeit ein Wert zugewiesen werden.

/BL-STR./ Block-struktur verletzt, z.B. IF .. THEN ... END; oder BEGIN ; FIN; Tritt häufig als Folgefehler auf.

Bilanzdetektierbare Fehler:
.....

MISSING PROC/TASK. Dem Compiler wurde durch SPECIFY vorgeschwindelt, daß die angegebenen PR/TASK weiter unten im Modul noch definiert werden. Nun - zum MODEND - wird dem Compiler klar, daß er nicht mehr hoffen darf.

MISSING LABELS. Die angegebenen Marken wurden zwar angesprochen, aber nicht oder nicht blockkonform definiert.

LIMIT-ERROR. Die tatsächliche Größe des Modules übersteigt den durch S=.. (bzw. default) festgelegten Kopfeintrag. Das Modul kann nur mit zusätzlichem SZ-Parameter beim LOAD geladen werden.

Nicht sprachbedingte Fehlerkonditionen.
.....

CAN'T-READ. Der Eingabefile produziert Lesefehler: Abbruch.
END-OF-SI. File zu Ende, bevor 'MODEND' akzeptiert wurde.
;...;TOO-LONG. Die Anweisung ist zu lang (mehr als 10 Zeilen)
INT. ERROR Bitte schreiben Sie uns, wenn klar ist, dass Ihre Hardware/E-PROMs etc. in Ordnung ist!!
OVERFLOW xy Nicht genügend Listenplatz, SZ bei P zu klein

Bei Auftreten eines der unten aufgelisteten Fehler wird die entsprechende Meldung ausgegeben und die verursachende PEARL-Task läuft nach den beschreibenen Aktionen weiter. Einige Fehler führen zu einer Terminierung oder einer Suspendierung der PEARL-Task, dies ist bei den entsprechenden Meldungen vermerkt.
Wenn das Programm mit der Markierungsoption /*+M*/ uebersetzt wurde (zumindest teilweise), so wird vor der Fehlermeldung die letzte registrierte Hochsprachzeilennummer ausgegeben. (Im Programmbereich mit abgeschalteter Markeroption werden keine neuen Nummern registriert!)

DV/0 Divided by zero.
Es wurde versucht, durch Null zu teilen. Für das Resultat wird die größte mögliche Zahl eingesetzt.

END-OF-FILE Das Ende eines Files ist beim Lesen überschritten worden. Der File wurde vor dem Lesen nicht auf den Anfang gesetzt (CALL REWIND(dation) vergessen).

FL.OV Floating overflow. ABS(result) is GT 1.1...E37.
Der Betrag des Resultates ist größer 1.1...E37.

INPUT-SYNTAX Input-character violates format-syntax.
Das Zeichen paßt nicht zum Eingabeformat, z.B. Buchstabe bei FIXED-Format.

ILL) Illegal ')' in FORMAT-> destroyed/wrong coding.
Die Anzahl rechter Klammern stimmt nicht. Kann nur bei Hyperprozessor-Benutzung im Assemblerprogramm passieren.

IOFM I/O-FORMAT does not conform to data-type in list.
Das Format paßt nicht zum Datentyp des Elements.

IONS I/O not set up -> Destroyed/wrong machine-code.
I/O nicht eröffnet! Kann nur bei zerstörtem Code oder Assemblerprogramm passieren. Terminiert die PEARL-Task.

NDSF No data-spec. in FORMAT. (PUT X TO .. BY SKIP;)
Die Formatanweisung für das Datum fehlt völlig.

NDUR Negative duration (AFTER ..., ALL ... etc.)
Die Zeitdauer ist negativ.

NIEX Negative input-exponent for 'FIXED' number.
Der Exponent für die Eingabe einer FIXED-Zahl ist negativ.

NIM-D0 Not implem. hyperproc-instruction. Opcode in D0. Die V-Number in D0 ist nicht implementiert. Alte S-Records geladen ?! Suspendiert die PEARL-Task.

OBIN Overflow B-formatted input. Die Zahl paßt nicht in das angegebene Format.

OEXI Overflow exponent on (numeric) input. Der Exponent ist zu groß oder zu klein.

OPNIDF Open by IDF ...Syntax or length error. Die Zeichenkette beim IDF ist fehlerhaft.

PNUM Parameter-numbers on caller-/proc-side not equal. Die Anzahl der Parameter auf der Aufruferseite stimmt nicht mit Anzahl der Prozedurseite überein. Terminiert die PEARL-Task.

PTYP Parameter-types on caller-/proc-side: no match. Der Parametertyp stimmt nicht überein. Terminiert die PEARL-Task.

PXFR Parameter xfer illegal. (IDENT-proc + INV-call) Der Aufruf stimmt nicht.Terminiert die PEARL-Task.

X/PAG- X-/PAGE-count negative. (A=-3 ....X(A),PAGE(A),.) Die Anzahl Blanks oder PAGE ist negativ.

PATHLT: TOO LONG in der angegebenen Pathlist wurde ein zu langer Bezeichner gefunden (Filename > 8 Zeichen).

READ Can't read from inputfile.(GET + file empty etc.) Es kann nicht vom Eingabefile gelesen werden.

RND/EN 'ROUND'/'ENTIER' overflow. Result > FIXED(15) Das Resultat einer Rundung paßt nicht in eine FIXED(15) Zahl.

SKP- SKIP-count is negative. (A=-3; ... SKIP(A),...) Die Anzahl Skips ist negativ.

SUSP: TASK NOT FOUND extern zu suspendierende Task wurde nicht gefunden. Tritt bei einem Fremd-Suspend auf.

TIME-OUT Time out error. Nur bei bestimmten I/O-Treibern: Zeitüberwachung angesprochen, Gerät antwortet nicht.

XIOV 'FIXED'-number input-overflow. Die Zahl paßt nicht in die FIXED-Zahl.

(OVF Left bracket or 'R'-FORMAT nesting overflow. Zuviele öffnende Klammern beim Format.

A L L G E M E I N E S.

Der UH-Assembler für den Mikrorechner MC68000 ist virtuell codiert, er verhält sich daher unabhängig von der Gastmaschine in allen Implementierungen in gleicher Art und Weise.

Ausgegeben werden 'S'-Rekords in einem gegenüber dem Original (Motorola) erweiterten Format. Die Ausgabe ist abgestimmt auf das PEARL-System RTOS-UH und dort bindefähig. Bei entsprechender Selbstbeschränkung (keine relativen Langadr., keine globalen Symbole) ist jedoch Motorolakompatibilität erreichbar.

Bei der Codierung richte man sich nach dem Hardware-reference Manual vom Motorola. Man beachte dabei, daß
---> die Immediateadressierung durch '=' statt '#' spezifiziert werden muß.

E I G E N S C H A F T E N.

Der Assembler benötigt 2 Durchläufe. Das Quellprogramm muß zweimal angeboten werden, wenn die Quelldatei nicht 'rückspulbar' ist, es sei denn, der Assembler arbeitet im 'automatic scratch-pad'-Mode, bei der er selbst eine Zwischendatei anfertigt. Die Betriebsparameter werden über das Bedieninterface besetzt:

CO=(Dev/File)	Senke für die S-Records.
LO=(Dev/File)	Senke für die Liste mit Zeilennummer und Hexcode. Auch bei LO=NO werden die fehlerhaften Zeilen aufgelistet.
SC=(Dev/File)	Scratch-device. Ist die Input - Datei rückspulbar, so muß SC nur angegeben werden, wenn eine neue Quellendatei erstellt werden soll. Wenn SC angegeben ist, so wird es in jedem Fall benutzt. Bei SC=NO muß bei nicht rückspulbaren Dateien der Quellcode zweimal angeboten werden.
SI=(DEV/FILE)	Eingabe-datei.

Bei Betrieb unter RTOS-UH kann zusätzlich noch der Speicherbereich und/oder die Bearbeitungspriorität angegeben werden.

A U F B A U D E R P R O G R A M M Z E I L E N

Jede Zeile enthält entweder einen Maschinenbefehl, einen virtuellen (Hyperprozessor) Befehl oder eine Direktive für den Assembler selbst. Mehrere Anweisungen pro Zeile sind nicht zugelassen. Als Feldtrennung wird die Lücke (mind. 1 Blank), bei Befehlen, die mit und ohne Operand auftreten können, ein Block > 10 Blanks benutzt.

L A B E L F I E L D

Das Feld ist leer (mind. 1 Blank) oder enthält ein max. 6 Zeichen langes 'Labelsymbol'. Solche 'Labels' können auf drei Arten definiert werden:

I. In Spalte 1 bedeutet '>': Das folgende Symbol wird als globales Symbol definiert und ist dem Lader später namentlich bekannt. Weitere Behandlung wie unter II.

II. In Spalte 1 steht ein Buchstabe oder das Buchstabenersatzzeichen '#'. Es wird ein max. 6 Zeichen langes Symbol definiert und ihm der Wert des Location-counter (REL oder ABS) zugewiesen.Danach darf ein ':' folgen, muß aber nicht - eine Feldtrennung genügt.

III. Spalte 1 enthält ein Leerzeichen. Das Symbol darf weiter hinten im Feld beginnen, aber es muß das Zeichen ':' folgen, damit erkennbar ist, daß kein OP-code gemeint sein kann.

Innerhalb des Symboles sind Ziffern erlaubt.

O P E R A T I O N S F E L D

I. HARDWARE-INSTRUKTIONEN.

Alle Mnemos des HARD-WARE-REFERENCE-MANUALS sind zugelassen. Eine Längenspezifikation (.L,.B,.W) ist nur hinter solchen Mnemos zugelassen, bei denen überhaupt eine Wahlfreiheit besteht. Fehlt die Längenangabe, so wird die dem Mnemo eingeprägte Länge - z.B. '.L' bei LEA - oder .W bei Wahlfreiheit substituiert. Bei dem Befehl MOVE muß daher im Ausnahmefall MOVE ...,USP die Länge MOVE.L ...,USP explizite angegeben werden, da sonst ein Length-error diagnostiziert wird. Bei Befehlen des Branch-types (BRA,BGE,BPL etc.) ist die Angabe .L oder .S möglich. Sie nimmt dem Assembler die Wahlfreiheit bei der Codierung dieser Befehle.

Alle Hardware-Befehle werden auf geraden Positionszählerstand gesetzt, so kann z.B. nach einem 'DC.B' vom Assembler automatisch ein nicht besetztes Byte eingefügt werden, wenn der nachfolgende Befehl eine gerade Adresse verlangt.

II. VIRTUELLE PEARL-LAUFZEIT-BEFEHLE (HYPERPROZESSOR)

Diese Instruktionen haben den folgenden Aufbau

```
V...   OP1,OP2,OP3,OP4
```

Dabei steht '...' für eine max. 3-stellige Dezimalzahl im Bereich 0...255. Es sind 0...4 Operanden möglich, deren Syntax z.T. von der der realen Befehle abweicht. (s.u.). Gleiches gilt bei Verwendung eines mit OPD.V definierten virtuellen Benutzermnemos.

III. ASSEMBLER-DIREKTIVEN.

Es gibt folgende Direktiven: ('AE','RE', ETC. s.u.)

```
      DC....            Datenablage wie bei Maschinenbefehlen,
      DC....            aber bei DC.B keine Positionsrundung.
      DC  'AE',...      Ablage der 16-Bit Daten (Wert 'AE').
      DC.W 'AE',...     Wie oben. (.W ist Defaultbesetzung).
      DC.L 'AE',...     Ablage von 32 Bit Daten mit Wert 'AE'.
      DC.L 'EG',...     Adr. externes globales Symbol ablegen.
      DC.B 'AE',...     Ablage der 8-Bit Daten ohne vorherige
                        Positionsrundung.
      DC.B 'str',..     Ablage des Textstrings 'str' mit fort-
                        laufenden Adressen. Das Zeichen ' ist
                        durch '' zu ersetzen.
                        Bsp:     DC.B  'AB''CD','''' => AB'CD'
                        Mischung  'STR','AE','STR',.. ERLAUBT.

      END               Ende dieser Assembliereinheit.Zwischen
                        'END' und Kommentar mind. 10 Blanks!
      END   'RE'        Rückbezug für RTOS-UH herstellen.

LBL   EQU   'AE'        Defin. von LBL mit Wert 'AE' absolut.
                        Die Eigenschaft 'logisch' wird nicht
                        durch das EQU übertragen. VORSICHT!
LBL   EQU   'RE'        Defin. von LBL mit Wert 'RE' relativ.
LBL   EQU   'RG'        Defin. von LBL als Registersymbol.
LBL   EQU    SR         Defin. von LBL als Status-register.
LBL   EQU   CCR         Defin. von LBL als Cond.-Coderegister.
LBL   EQU   USP         Defin. von LBL als USER-A7.

MNE   OPD   'AE'        Defin. des Nutzer-Mnemos (ohne Opera-
                        nden) MNE durch das 16-Bit-Wort 'AE'.
MNE   OPD.V 'AE'        Defin. des Nutzer-Mnemos MNE als vir-
                        tueller Laufzeitbefehl. 'AE' muß im
                        Bereich 0...255 liegen, da sonst bei
                        der späteren Benutzung ein 'LIMIT'-
                        error erzeugt werden kann. Der so de-
                        finierte Befehl kann 0 ... 4 Operan-
                        den haben und und unterliegt der bei
                        den V...- Befehlen üblichen Syntax.
      ORG   'AE'        'AE' muß in Pass 1 definiert sein,
                        der Wert wird in den Positionszähler
                        geladen. Das Programm ist nun im ABS-
                        mode und kann vom RTOS-UH-Lader nicht
                        korrekt geladen werden, sofern nicht
                        ausschließlich relativ adressiert
                        wird.
      RORG 'AE' or 'RE' Wie bei ORG jedoch ist das Module
                        jetzt verschieblich und kann korrekt
                        vom Relativlader geladen werden.
      PAGE              New Page. PAGE steht auf dem Anfang.

      DS    'AE'        Define storage, mit 'AE' angegebene
                        Zahl von Bytes freihalten. Inhalt der
                        Bytes nach dem Laden undefiniert.
```

O P E R A N D E N - F E L D

Die Adressierungsarten werden kleinlich mit den Hardwaremöglichkeiten verglichen. Diagnose:'MODE-ERROR'.

Bei 'MOVEM' ist eine Ordnung der Registerliste von D0 in in Richtung A7 zwingend vorgeschrieben. (Schutz vor Tippfehlern).

```
        ADR.ART      SYNTAX              BEMERKUNGEN

        ABS-SHORT    'AE'            'AE' siehe 'ABS EXPRESSion'.
        ABS-LONG     'AE'            Wenn 'AE' in Pass 1 > $FFFF.
   (V)               'AE'.L          Wertunabhängig LONG.
   (V)  Ext.glob.    'EG'            Wie ABS-LONG, kein Ausdruck.
   (V)  REL-LONG     'RE'.L          Rel. durch Lader. Wie 'AE'.L
        REG.DIR.     'RG'           'RG'=Register-Symbol.
        INDIR.REG.   ('AR')         'AR'=Adreßregistersymbol.
        PREDECR.     -('AR')                 ''
        POSTINCR.    ('AR')+                 ''
        IND.REG+DISP.'AE'('AR')    'AE' muß in 16 Bit passen.
        IND.A4+DISP. 'AE'.T        TASK-WSP-Adr.-mode.
   (V)  IND.A5+DISP. 'AE'.X        Local WSP Adr.-mode.
    V   IND(IND.A5+DISP) 'AE'.Z    32 Bit-end-adr. nur bei V!
        DSP+AR+IND.  'AE'('AR','RG')  .W-index, 'AE' 8 Bit!
           ''        'AE'('AR','RG'.W)          ''
           ''        'AE'('AR','RG'.L) .L index,    ''
   (V)  PC-REL.      'RE'           'RE'=REL.Expression, s.u.
        PC-REL+IND.  'RE'('RG')     .W index
           ''        'RE'('RG'.W)        ''
           ''        'RE'('RG'.L)  .L Index
        LABEL        'AE'           Im ORG-Mode. Bcc, BRA, DBcc
         ''          'RE'           Im RORG-Mode. ''   ''   ''
        IMMEDIATE    ='AE','RE'     ACHTUNG: '=' statt '#'!
```

(V) Die mit '(V)' gekennzeichneten Adressierungen sind auch bei virtuellen Befehlen zugelassen.

A U S D R Ö C K E :

'AE' ist ein Ausdruck, dessen Wert lageunabhängig ist
'EG' ist ein Bezug auf ein extern definiertes Symbol:
'EG' := >symbol; oder 'EG':= >symbol+$x;
Für x ist eines der Zeichen 0...F (als Offset) erlaubt. 'symbol' repraesentiert das übliche max. 6 Zeichen lange Symbol.

BEISPIEL: JMP >TEST+$2

Im eigenen Block definierte Globalsymbole können zwar, sollten aber nicht über diese Konstruktion angesprochen werden. (Um den Lader zu entlasten).

'RE' ist ein Ausdruck, dessen Wert einfach positiv von der Lage des Programmes abhängig ist.

In beiden Typen von Ausdrücken sind Klammerungen erlaubt, ebenso die Voranstellung monadischer Operatoren (+ oder -). Ausdrücke werden grundsätzlich mit 32-bit Arithmetik berechnet und erst in Pass 2 auf Einhaltung der zulässigen Grenzen überprüft.

Bei 'AE' in Immediate und 'DC'-Anwendungen wird zusätzlich zwischen 'logisch' und 'arithmetisch' unterschieden. Ein 'AE' ist 'logisch' wenn in ihm mindestens eine Sedezimalzahl auftritt. Eventuell muß man also die Zahl $0 addieren. Damit ist z.B. 40126+$0 auch bei Beschränkung auf 16 bit legal. (Da 'logisch').

Elemente in Ausdrücken:

Dezimalzahlen	Z.B. 24,108637 (Immer als 'AE')
Sedez.zahlen	Z.B. $0,$AFFE,$2CDE3 ''
Nonregsymbol	Z.B. X,AB23, (Max. 6 Zeichen)
''	Können 'RE' oder 'AE' sein.
Loc. counter	$ Ohne folgende Ziffern, ist 'RE' im RORG-mode sonst 'AE'.

Dyadische Operatoren:

+	Addition. ABS+ABS=ABS, ABS+REL=REL, REL+ABS=REL, REL+REL nicht erlaubt!
-	Subtraktion, REL-REL=ABS, REL-ABS=REL, ABS-ABS=ABS, ABS-REL nicht erlaubt!
*	Multiply, nur bei zwei 'AE' erlaubt. Es wird in signed 32 bit arithm. gerechnet.
/	Divide. Wie bei Multiply, nur 'AE' etc.

Die Prezedenz der Operatoren ist wie üblich, d.h. *,/ geht vor +,-. Durch Klammerung wird die Prezedenz übersteuert.

T A B E L L E N K A P A Z I T Ä T . .

Unser Assembler benötigt für jedes Symbol 14 Bytes Listenplatz. Von dem angebotenen Workspace (z.B. SZ=xx unter RTOS-UH) gehen zusätzlich noch einmal ca. 600 Bytes für Pufferung etc. verloren. So kann mit z.B. SZ=6000 ein Programm bis über 1700 Symbolen übersetzt werden.

This page intentionally left blank.

F E H L E R - D I A G N O S E .

Fehlermeldungen werden zeilenweise eingebettet und durch einen Stern unter dem inkriminierten Zeichen markiert, an der Stelle,an der der Assembler die Abweichung erkennen konnte.

Im Fehlerfalle wird wann immer möglich ein 3*NOP-Code eingesetzt um ggf. nach dem Laden korrigieren zu können. Dies ist allerdings bei bestimmten Fehlern, die erst im Pass 2 erkannt werden können, nicht möglich.

Ein vorzeitiger Abbruch erfolgt bei der TAB.-OVERFL. - kondition.(s.u.)

'DEF-ERROR'	Falsch definiert. z.b. Vorwärtsbezug bei EQU oder zu spät als Register, d.h. nachdem das Symbol bereits benutzt wurde.
'DOUBLE-DEF'	Das Symbol wurde mehrmals definiert.
'FORW.REF.'	Vorwärtsbezug hier nicht erlaubt.
'LENGTH'	Operation und Länge harmonieren nicht.
'LIMIT'	Grenzwert des Ausdrucks überschritten, oder z.B. durch Null dividiert.
'MODE'	Adressierungsart ist hier nicht erlaubt.
'P1/2-MATCH'	Zusammenbruch. Label im Pass 2 entdeckt, das im Pass 1 noch nicht vorkam.
'R/A-ERROR'	Ausdruck 'RE' statt 'AE' oder umgekehrt. Der falsche Typ wird jedoch eingesetzt, Programm i.a. unbrauchbar.
'SYNTAX'	Keine Produktionsregel gefunden. (3*NOP).
'UNDEF'	Das Symbol wurde nicht definiert.
***TAB-OVERFL	Der Platz reicht nicht aus. Mit SZ beim Aufruf des Assemblers Listenplatz vergrössern (bis SZ=7FFE möglich!).
INP.FAIL	Abbruch des Laufes, weil Input-file nicht lesbar/vorhanden ist, oder die End-Of-File Bedingung vor dem END eingetreten ist.

This page intentionally left blank.

RTOS-UH eignet sich zusammen mit der Vielzahl von Systemtraps auch sehr gut für die maschinennahe Codierung. Dabei sind allerdings einige Konventionen zu beachten, da sonst schon evtl. unmittelbar nach dem Laden ein 'Absturz' des gesamten Systemes verursacht werden kann. Der Lader benötigt nämlich für Module einen sog. 'Modulkopf' und für Tasks einen sog. 'Taskkopf' im Vorspann des eigentlichen Programmes. Der 'Taskkopf' ist dabei ein verlängerter 'Modulkopf'. Besteht ein Programmblock aus mehreren Modulen/Tasks, so müssen diese vom Programmierer miteinander verzeigert werden. Der 'Nullzeiger' zeigt an, daß es keinen Vordermann oder Hintermann zu diesem M/T-kopf gibt.

```
MODULKOPF:          DC.L   0    or: address of next  M/T-head
---------           DC.L   0    or: address of last  M/T-head
                    DC     $0010       Typeindicator: 'Module'
                    DC.B   '......'    6 bytes name-link (s.u.)

            -->     von hier  ab  freie  Kodierung des Modules.

TASKKOPF:           DC.L   0    or: address of next  M/T-head
---------           DC.L   0    or: address of last  M/T-head
                    DC     tasktype         (ist unten erläutert)
                    DC.B   '......'    6 bytes name-link (s.u.)
                    DC     PRIO   Defaulttaskpriorität 1...255
                    DC.L   Workspacelänge d. Task: mind. 120 !!
                    DC.L   0,0     Zeiger f. systemeigene Zwecke
                    DC     PRIO   ist  später  variable Laufprio
                    DC.L   Startadresse  Adr. 1. Anweisung Task
                    DC.L   0,0,0,0,0,0,0,0,0,0,0     (10x4)
                    DC.L   0,0,0,0,0,0,0      total 64 bytes zero

          -- >       von hier an freie Kodierung der Task.
```

'namelink': Entweder unmittelbar (6 ASCII-Char.) der Task- bzw. Modulname oder in den ersten 4 Byte steht die auf die Adr. von 'namelink' relativierte adresse des (beliebig langen) Textes (mit $FF beendet), der den Modul-/taskbezeichner darstellt. Im 2. Fall sind die Bytes 5 und 6 ohne Bedeutung, müssen aber angelegt werden.

'tasktype':

$0001	=	Normale Task, ohne 'RESIDENT'-Attribut.
$0081	=	'Residente' Task, die ihren TWSP behält.
$0041	=	Autostarttask, läuft nach Abort sofort.
$00C1	=	Kombination: 'Residente' Autostarttask.

HINWEIS: Der Taskkopf hat sich nach Aufnahme der 68020-Familie wegen des Floatkoprozessors gegenüber früher geändert, es sind jetzt 24 Nullbytes mehr als früher (als Minimum) erforderlich. Unbedingt also alte Programme hier ändern. Die längeren Köpfe laufen natürlich auch auf alten RTOS-UH Versionen.

!!

This page intentionally left blank.

TABLE of TRAPS inside RTOS-UH

Umklammerte Traps sind nicht im Nukleus angesiedelt sondern in irgendeiner anderen Scheibe angesiedelt - sofern vorhanden.

ACHTUNG: Traps niemals auf Interrupt-Ebene aufrufen! *

$4E40	ACTQ	Activate quick. Task-ID-pointer is in A1. *
$4E41	TERMI	Exit = Terminate internal = self-termination *
$4E42	CON	Continue task given by name in $66(A4) *
$4E43	QDPC	Start a dispatching cycle *
$4E44	----	Reserved *
$4E45	SCAN	System-scanner (for mounting of RTOS-UH + loader)*
$4E46	REQU	Request semaphore. Adr. of sema is in A1 *
$4E47	RELEA	Release semaphore. Adr. of sema is in A1 *
$4E48	FETCE	Fetch a communication-element, D1.L=size, A1=ptr *
$4E49	RELCE	Release a communication-element. Pointer is in A1*
$4E4A	XIO	Xfer a communication-element to in/output-handler*
$4E4B	PENTR	Procedure entering. (Workspace alloc. etc.) *
$4E4C	RETN	Return from procedure. (complement to PENTR) *
$4E4D	TOQ	Take of queue. (Inside i/o-handler-tasks) *
$4E4E	(TOV)	Hyperprocessor 'on' = to virtual code switching *
$4E4F	OFF	Dispatching and interrupts 'off' + supervisormode*
$A000	TERME	Terminate (external) task by name in $66(A4) *
$A002	ERROR	Send error-message to corresponding userterminal *
$A004	WSFS	Workspace forward search. (A1 is loaded) *
$A006	ITBO	Identify task by name in $66(A4). (A1 is loaded) *
$A008	WSFA	Workspace fixed address request. *
$A00A	IOWA	I/O-wait by communication-element in A1. *
$A00C	WSBS	Workspace backward search. (A1 is loaded) *
$A00E	GAPST	Generate and prepare a subtask (son-process) *
$A010	TERV	Terminate (self) and vanish. (son-process exit) *
$A012	DVDSC	Device-(facility)-tester. (LDN expected in D1) *
$A014	ACT	Activate task by name in $66(A4) *
$A016	TIAC	Time-scheduled activation of task by name $66(A4)*
$A018	TICON	Time-scheduled continuation '' *
$A01A	ACTEV	Interrupt-scheduled activ. of task by name '' *
$A01C	CONEV	Interrupt-scheduled cont. of task by name '' *
$A01E	QSA	Quote-scanner with answer. (Lex.text by adr. A2) *
$A020	RUBBL	Rubber for blanks. (Text-pointer is in A2) *
$A022	PREV	Prevent task by name in $66(A4) *
$A024	TIACQ	Time-scheduled activ. of task (quick) by TID = A1*
$A026	TRIGEV	Trigger = simulation of an interrupt *
$A028	SUSP	Self suspending of executing task. *
$A02A	RWSP	Release workspace by pointer in A1. *
$A02C	TIRE	Time-scheduled RESUME of a task *
$A02E	(PIT)	Process-data input. (implement. dependent) *
$A030	(POT)	Process-data output. '' *
$A032	ENAB	Enable selected process-interrupts. *
$A034	DISAB	Disable selected process-interrupts. *
$A036	LITRA	Line-tracer in real environment. *
$A038	LITRAV	Line-tracer virtual (inside hyperproc) *
$A03A	CSA	Character scan alternatively. (text ptr in A2) *
$A03C	IMBS	Identify module by string. (String ptr in A2) *

```
$A03E  RCLK     Read system-clock. Result is in D1.L                 *
$A040  ITS1T    Index-tester for 1-dim arrays.                       *
$A042  ITS2T    Index-tester for 2-dim arrays.                       *
$A044  ITS3T    Index-tester for 3-dim arrays.                       *
$A046  MD2B60   Multiply D2.L by 60  (long + fast!)                  *
$A048  ITBS     Identify task by string. (string-ptr in A2)          *
$A04A  ----     Reserved                                             *
$A04C  INTD1    Integer (long) into D1 by text-pointer in A2         *
$A04E  TICONQ   Time-scheduled cont. of task (quick) by TID=A1       *
$A050  CONQ     Continue task quick by TID=A1.                       *
$A052  DELTST   (Right) Delimiter-test of text. (ptr is A2)          *
$A054  PREVQ    Prevent task quick by TID=A1.                        *
$A056  EVACTQ   Interrupt-scheduled task-activ. quick by TID=A1      *
$A058  TERMEQ   Terminate task quick by TID=A1.                      *
$A05A  EVCONQ   Interrupt-scheduled task-cont. quick by TID=A1       *
$A05C  -----    RESERVED.                                            *
$A05E  STBCLK   Set Battery Hardware-Clock                           *
$A060  ITS1TL   Index-tester for 1-dim arrays with long index.       *
$A062  ITS2TL   Index-tester for 2-dim arrays with long index        *
$A064  ITS3TL   Index-tester for 3-dim arrays with long index        *
$A066  (DATASC)  Date to ASCII conversion                            *
....              ''                                                 *

$A07E  -----    RESERVED.                                            *
                                                                     *
```

Die folgenden Line-A Traps können in besonderen Anwenderimplementierungen frei belegt werden. Die Transfer-adressen aller Line-A-traps beginnen auf $400 (für $A000), $404 (für $A002) .. usw. in 4-byte-Schritten. Also: Rechtes Trap-byte mal 2 plus $400. Der Anschluß kann dann etwa über Scheibe 14 hergestellt werden. Dabei ist zu beachten, daß von $800 nach unten der System-stack den Links entgegenwächst und diese ggf. zerstört werden.

```
$A080  ???      User-applicable for special implementations          *
....   ....                          ''                              *

$A0FE  ???      Last allowed user-trap.                              *
```

Desweiteren steht noch eine Subroutine zur Verfügung (schneller Anschluß):

```
CD7TAS          Convert D7 to ASCII-String                           *
```

Activate Task by name. A C T = A 0 1 4

Eingaberegister: D1.W Priorität der Aktivierung.
$66(A4) 6 ASCII-bytes des Tasknamens oder 4 byte Adresse des Namensstring, der mit $FF endet.

Veränderte Register: D7,D1 Bei Fehler auch D5,D6.

Funktionsbeschreibung: Die angegebene Task wird in der Speicherverwaltung gesucht. Dabei wird die Usernummer der Tasks nicht berücksichtigt. Falls eine so bezeichnete Task nicht gefunden wird, so erfolgt Fehlermeldung und eine Operation unterbleibt.

Falls D1.W EQ 0, wird D1 aus der Taskdefaultprio geladen. Ist D1 negativ, so erfolgt eine Fehlermeldung und die Aktivierung unterbleibt. Danach wird geprüft, ob die Task bereits im Dispatcherring steht. Ist dies nicht der Fall, so wird sie gemäß der Priorität aus D1 eingelinkt und der Trap endet mit Dispatcherstart.

War die Task dagegen bereits im Dispatcherring, so wird die Blokkierbedingung 'waiting for activation' untersucht und gelöscht.

a) Blockierbedingung war gesetzt.
Es wird geprüft, ob die Task prioritätsgerecht eingelinkt ist, falls nicht, so wird sie entsprechend 'umgelinkt'. Der Trap endet mit einem Dispatcherstart.

b) Blockierbedingung war nicht gesetzt.
Die Aktivierung wird mit ihrer Priorität in den Puffer der Task geschrieben, falls dort noch Platz ist. (Max. 3) Wenn kein Platz im Puffer ist, so erfolgt Fehlermeldung und die Operation unterbleibt. Der Trap endet ohne Dispatcherstart.

Fehlermeldungen:
....WRONG PRIO. (D1.W negativ)
....OVERFLOW (ACT). (Aktiv.puffer Überlauf)
....NOT LOADED (ACT) (Task wurde nicht gef.)

Eingaberegister: D1.W Priorität
OPNAME.T Textadresse oder Text
OPFATI.T Prozeß-Interrupt-Ereignis
Ausgaberegister: -
Veränderte Register: D1,D7,A1,A2

Funktionsbeschreibung:

Eine Task, deren Name oder deren Adresse des Namens in OPNAME.T = $66(A4) steht, wird zur Aktivierung eingeplant. In D1 wird die Priorität der Aktivierung übergeben. Ist D1 gelöscht, wird die Default-Priorität eingetragen. Die aktuelle Priorität der laufenden Task bleibt unbeeinflußt. In OPFATI.T = $6C(A4) wird die Prozeß-Interrupt-Maske eingetragen, auf die die Task eingeplant werden soll. Bestehende Einplanungen werden gelöscht. Die Fehlermeldungen entsprechen denen beim Trap ACT beschriebenen.

Beispiel:

```
ACTEV    OPD  $A01A            Trap-Definition
         ...
         LEA     TSKNAM,A0     Adresse des Tasknamens
         MOVE.L  A0,OPNAME.T   Eintrag der Adresse
         CLR     OPNAME+4.T    kein Text
         MOVE.L  =$80000000,OPFATI.T  Interrupt-Maske
         ACTEV                 auf Prozeßinterrupt einplanen
         ...
```

Activate Quick

A C T Q = 4 E 4 0

Eingaberegister: D1.W Priorität
A1.L Adresse der Task (TID)
Ausgaberegister: -
Veränderte Register: D1,D7,A1,A2

Funktionsbeschreibung:

Eine Task, deren Adresse in A1 steht, wird aktiviert. In D1 wird die Priorität der Aktivierung übergeben. Ist D1 gleich Null, wird die Task mit ihrer Standardpriorität gestartet. Negative Prioritäten sind den Systemtasks vorbehalten und damit nicht erlaubt, sie führen zu einer Fehlermeldung. Läuft die Task schon, wird die Aktivierung gepuffert. Beim Überlaufen des Aktivierungspuffers erhält man ebenfalls eine Fehlermeldung.

Beispiel:

```
ACTQ    OPD  $4E40      Trap-Definition
        ....            TID in A1
        MOVE   =$20,D1  Prio=$20
        ACTQ            Activate Task
        ....
```

Convert D7 to ASCII-String

kein TRAP

Eingaberegister: D7.W Hex.-Zahl
Ausgaberegister: D6.L ASCII-Text der Hex.-Zahl
Veränderte Register: -

Funktionsbeschreibung:

Hierbei handelt es sich nicht um einen Trap! Vielmehr kann diese Routine über die Adresse

```
JSR    CD7TAS+JUMTBL
```

angesprungen werden. Die Hex.-Zahl in D7.W wird in einen ASCII-String verwandelt, der in D6.L steht. Diese Routine kann genutzt werden, um sich die Umwandlung in ASCII-Strings z.B. bei der Ausgabe einer Adresse zu sparen. D7.W ist nach der Routine unverändert.

Beispiel:

```
JUMTBL    EQU  $8F4        Start der Jumptable
CD7TAS    EQU  $6          Offset
          ...
          MOVE  =$12A4,D7      Hex.-Zahl
          JSR  CD7TAS+JUMTBL   konvertieren
          ...                  D6=$31324134
```

Continue

C O N = 4 E 4 2

Eingaberegister: OPNAME.T Textadresse oder Text
Ausgaberegister: -
Veränderte Register: D7,A1,A2

Funktionsbeschreibung:

Eine Task, deren Name oder deren Adresse des Namens in OPNAME.T = $66(A4) steht, wird fortgesetzt. Ist die Task nicht geladen oder nicht suspendiert, wird eine entsprechende Fehlermeldung ausgegeben und der Aufrufer suspendiert.

Beispiel:

```
CON     OPD  $4E42          Trap-Definition
        ...
        LEA     TSKNAM,A0   Adresse des Tasknamens
        MOVE.L  A0,OPNAME.T Eintrag der Adresse
        CLR     OPNAME+4.T  kein Text
        CON                 fortsetzen
        ...
```

Continue by interrupt-schedule — C O N E V = A 0 1 C

Eingaberegister: OPNAME.T Textadresse oder Text
OPFATI.T Prozeß-Interrupt-Maske
Ausgaberegister: -
Veränderte Register: D7,A1,A2

Funktionsbeschreibung:

Eine Task, deren Name oder deren Adresse des Namens in OPNAME.T = $66(A4) steht, wird zur Fortsetzung eingeplant. Bestehende Einplanungen werden gelöscht. In OPFATI.T = $6C(A4) muß die Prozeß-Interrupt-Maske eingetragen sein.

Beispiel:

```
CONEV     OPD  $A01C            Trap-Definition
          ...
          LEA     TSKNAM,A0     Adresse des Tasknamens
          MOVE.L A0,OPNAME.T    Eintrag der Adresse
          CLR     OPNAME+4.T    kein Text
          MOVE.L =$00040000,OPFATI.T  Interrupt-Maske
          CONEV                 bei Interrupt fortsetzen
          ...
```

Continue quick

C O N Q = A 0 5 0

Eingaberegister: A1.L Adresse der Task (TID)
Ausgaberegister: -
Veränderte Register: D7,A1,A2

Funktionsbeschreibung:

Eine Task, deren Adresse in A1 steht, wird fortgesetzt. Ist die Task nicht suspendiert, wird eine entsprechende Fehlermeldung ausgegeben und der Aufrufer suspendiert.

Beispiel:

```
CONQ    OPD  $A050        Trap-Definition
        ...               TID in A1
        CONQ              fortsetzen
        ...
```

Character-scan alternate

C S A = A 0 3 A

Eingabe-Register: A2.L Adresse des zu unters. Textes.
Ausgabe-Register: A2.L Inkrementiert um 1 falls gefunden.
Veränderte Register: D7.

Funktionsbeschreibung:

Die beiden hinter dem Trap im Speicher folgenden Bytes werden nacheinander mit dem Zeichen auf (A2) verglichen. Stimmt eines der beiden mit dem Eingabetext überein, so wird A2 um eins erhöht und das Statusregister auf 'EQ' gesetzt. Stimmt keines der beiden Bytes mit (A2) überein, so bleibt A2 unverändert und im Statusregister wird die Kondition 'NE' gesetzt.

In jedem Fall werden die beiden auf den Trap folgenden Bytes bei der Rückkehr übersprungen.

Der Trap eignet sich für eine einfache Textanalyse und wird innerhalb des Bedieninterpreters eingesetzt. Er ist darum auch zur Realisierung neuer Bedienbefehle optimal geeignet.

BEISPIEL:

```
CSA     OPD   $A03A       Trap-definition
        .....            A2 zeigt auf Eingabetext
        CSA              Aufruf
        DC.B  'Aa'       Prüfe, ob kleines oder großes A folgt.
        BEQ   Irgendwo   Wenn ja, springe mit erhöhtem A2 nach
                         Irgenwo
```

Date to ASCII Conversion

D A T A S C = A 0 6 6

Eingaberegister:	D0.W	Datum
	A2.L	Zieladresse des Ausgabestrings
Veränderte Register:	D0.L	zerstört
	A2.L	zeigt auf erste freie Byte nach Ausgabestring
	SR	

Funktionsbeschreibung:

Die in D0 übergebene Datumsangabe (Anzahl der Tage seit 31-12-1984) wird in einen Ausgabestring (10 Zeichen, Aufbau: tt-mm-jjjj) verwandelt und auf die in A2 übergebene Adresse geschrieben. A2 wird um die Anzahl der Zeichen erhöht.
Ist D0.W = $0000 (nicht gesetztes Datum), so wird die Zeichenfolge '----------' ausgegeben.

Beispiel:

```
DATASC OPD    $A066       Trapdefinition
DATE   EQU    $80A        Datum (Systemzelle)
       ....
       MOVE   DATE,D0     Datum aus Systemzelle
                          laden
       LEA    BUFFER,A2   Zieladresse laden
       DATASC             Datum auf (A2)+
       ....
```

Eingaberegister:	A2.L	Adresse des zu unters. Textes
Ausgaberegister:	A2.L	Inkrementiert bis Delimiter
Veränderte Register:	SR	

Funktionsbeschreibung:

Es wird geprüft, ob der zu untersuchende Text als nächstes Zeichen einen Delimiter enthält. Als Delimiter gelten Semikolon($3B), Bindestrich($2D) und <Return>($0D). Blanks($20) und Kommata($2C) werden überlesen. A2 wird solange erhöht, bis ein Zeichen gefunden wird, das kein Blank oder Komma ist. Ist dieses Zeichen ein Delimiter, so wird das Statusregister auf 'EQ' gesetzt, andernfalls auf 'NE'. A2 zeigt auf dieses Zeichen.

Dieser Trap eignet sich zur Analyse eines Textes, z.B. bei der Realisierung neuer Bedienbefehle.

Beispiel:

```
DELTST   OPD  $A052
         .....          A2 zeigt auf den Eingabetext
         DELTST         Aufruf
         BNE  NOLIM     Springt, wenn kein Delimiter
         .....
```

Disable Prozeßinterrupt — D I S A B = A 0 3 4

Eingaberegister: D0.L Interruptmaske
Ausgaberegister: -
Veränderte Register: D0,D7

Funktionsbeschreibung:

Das Bitmuster in D0 wird so mit der Enable-Mask verknüpft, das alle Prozeßinterrupts, deren Bits in D0 auf '1' gesetzt sind, gesperrt werden.

Beispiel:

```
DISAB     OPD  $A034      Trap-Definition
          .....
          MOVE.L =$40000000,D0  Nummer des Interrupts
          DISAB           Sperrung des Intrrupts
          .....
```

Eingaberegister: D1.B LDN
Ausgaberegister: D1.L Diff. Adr. der Device-Parameter
Veränderte Register: D7

Funktionsbeschreibung:

Der Trap liefert die Differenz der Adresse der Device-Parameter zu A1, wie sie beim DD- und SD-Kommando beschrieben werden. In D1.B muß die LDN der Datenstation übergeben werden. Es wird nicht geprüft, ob die entsprechende LDN überhaupt im System vorhanden ist. Bei Veränderungen der Device-Parameter muß also sichergestellt sein, daß die Adresse einer gültigen LDN verwendet wird, sonst kann es zu Systemabstürzen kommen, die sich u. U. erst später zeigen.

Beispiel:

```
DVDSC     OPD  $A012       Trap-Definition
          ...
          MOVE.B =2,D1     LDN = 2 (Port 2)
          DVDSC            Adresse von Device-Para
          MOVE.B =1,1(A1,D1.L)   Zweites Byte von
          ...              A2: auf ESC-Sequenzen setzen
          ...
```

Enable Prozeßinterrupt

E N A B = A 0 3 2

Eingaberegister: D0.L Interruptmaske
Ausgaberegister: -
Veränderte Register: D7

Funktionsbeschreibung:

Das Bitmuster in D0 wird mit der Enable-Mask 'geodert'. Damit werden alle Prozeßinterrupts, deren Bits in D0 auf '1' gesetzt sind, freigegeben.

Beispiel:

```
ENAB    OPD  $A032        Trap-Definition
        .....
        MOVE.L =$80000000,D0  Nummer des Interrupts
        ENAB              Freigabe des Intrrupts
        .....
```

Write Error-message

E R R O R = A 0 0 2

Eingabe-Register: Parameter über PC, ggf. A1. sowie $66(A4)
Ausgabe-Register: - -
Veränderte Register: D7, PC (überspringt Wort nach dem Trap)

Funktionsbeschreibung:

Es wird eine wortweise zusammengesetzte Meldung auf das Terminal des verantwortlichen Nutzers geschrieben. Der Text wird durch den Inhalt der Zelle OPNAME.T = $66(A4) sowie durch das Wort hinter dem Trap bestimmt. Wenn dieses aus den Hexziffern 'abcd' besteht, so bestimmen b,c und d den eigentlichen Text, während a eine Zusatzinformation festlegt. In OPNAME.T steht entweder ein 6 Byte langer Text oder eine 4 Byte Adresse gefolgt von einem Nullwort (Adressindikator). Textende durch $FF. Der Text muß konstant sein, da er nicht gepuffert wird!

```
Bsp.      ERROR    OPD   $A002
          .. .
                   ERROR              Trap-aufruf
                   DC      $1234      a=1,b=2,c=3,d=4
          *  Meldung wäre (s.u) >>task:filename WRONG PRIO (XIO)
```

a: Enthält funktionelle Bits:

1	Ergänze den durch A1 ermittelbaren File-namen.
2	Unterdrücke den Text in OPNAME.
4	Z.zt. nicht belegt.
8	Suspendiere die aufrufende Task.

b: Auswahl aus folgendem Vorrat:

0:	Leerstelle	1:	NOT	2:	WRONG
3:	ZERO-DIV	4:	CHK	5:	BLOCKS
6:	BREAKPOINT	7:	DIRECTORY	8:	DISC
9:	MEMORY	A:	MODULE	B:	MISSING
C bis F:	undef.				

c: Auswahl aus folgendem Vorrat:

0:	leer	1:	BUS-ERROR	2:	LDN
3:	PRIO	4:	LOADED	5:	SUSPENDED
6:	ACTIVE	7:	COMMAND	8:	ADDRESS
9:	OP-CODE	A:	PRIVILEDGED	B:	OVERFLOW
C:	IN SYSTEM	D:	I/O	E+F:	undef.

d: Auswahl aus folgendem Vorrat

0:	leer	1:	(ACT)	2:	(TERMI)
3:	(CONTINUE)	4:	(XIO)	5:	(TRAP)
6:	(FLOPPY)	7:	LOADER-INPUT	8:	REC-CHECKSUM
9:	LABEL	A:	(MODE)	B:	TIMING
C:	INDEX	D bis F:	undefiniert.		

Die Ausgaben erledigt die Systemtask '#ERROR', falls vorhanden.

Interrupt-schedule activation quick E V A C T Q = A 0 5 6

Eingaberegister: D1.W Priorität
A1.L Adresse der Task (TID)
OPFATI.T Prozeß-Interrupt-Maske
Ausgaberegister: -
Veränderte Register: D1,D6,D7,A1,A2

Funktionsbeschreibung:

Eine Task, deren Adresse in A1 steht, wird zur Aktivierung eingeplant. Bestehende Einplanungen werden gelöscht. In D1 wird die Priorität der Aktivierung übergeben. Ist D1 gelöscht, wird die Default-Priorität eingesetzt. Die Priorität der aktuell laufenden Task wird nicht geändert. In OPFATI.T = $6C(A4) muß die Prozeß-Interrupt-Maske eingetragen sein.

Beispiel:

```
EVACTQ    OPD  $A056        Trap-Definition
          ...               TID in A1
          MOVE.L =$80004000,OPFATI.T  Interrupt-Maske
          EVACTQ            auf Interrupt einplanen
          ...
```

Event continue quick — E V C O N Q = A 0 5 A

Eingaberegister: A1.L Adresse der Task (TID)
OPFATI.T Prozeß-Interrupt-Maske
Ausgaberegister: -
Veränderte Register: D6,D7,A1,A2

Funktionsbeschreibung:

Eine Task, deren Adresse in A1 steht, wird zur Fortsetzung eingeplant. Bestehende Einplanungen werden gelöscht. In OPFATI.T = $6C(A4) muß die Prozeß-Interrupt-Maske eingetragen sein.

Beispiel:

```
EVCONQ    OPD  $A05A        Trap-Definition
          ...               TID in A1
          MOVE.L =$80001000,OPFATI.T  Interrupt-Maske
          EVCONQ            auf Interrupt einplanen
          ...
```

Eingaberegister: D1.L
Ausgaberegister: A1.L
Veränderte Register: D1,D6,D7

Funktionsbeschreibung:

Der Inhalt von D1.L wird als Größe des effektiv nutzbaren zu schaffenden Puffers im CE angesehen. Will man den Puffer (IOBUF, siehe Beschreibung CE in Kapitel E-III) nicht benutzen, so ist durchaus ein Aufruf mit D1.L=0 sinnvoll. Im Register A1 wird die Adresse des CEs zurückgegeben. Die Ausführung dieser Instruktion kann die exekutierende Task blockieren:

Fall a) Die Task hat bereits ihr Kontingent an CE-Speicherraum verbraucht. Sie wird mit 'CWS?' blockiert und erst wieder lauffähig, wenn andere in ihrem Besitz befindliche CEs zu freiem Speicher rückverwandelt sind.

Fall b) RTOS-UH hat nicht mehr genügend Speicher zur Verfügung. Die Task wird in der Kondition 'PWS?' blockiert und erst wieder lauffähig, wenn irgendwo genügend Speicher freigeworden ist.

Die Blockierung im Fall a) läßt sich vermeiden, wenn das least-significant Bit (ungerade Zahl) im Eingaberegister D1.L gesetzt ist. Dieses Bit wird nicht bei der Größenberechnung berücksichtigt, sondern dient als Indikator für die Zulassung einer 'kontingentüberschreitenden' Anforderung.

Das CE wird vom FETCE mit folgenden Einträgen vorparametriert:

PRIO	Eigenpriorität der Task
BUADR	Adresse von IOBUF(A1)
STATIO	$00
FNAME	8 Blanks

Die restlichen Parameter müßen von der Task besetzt werden, bevor das CE über 'XIO' benutzt werden kann. Natürlich dürfen auch die obigen Parameter (mit Ausnahme von STATIO) verändert werden.

Eingabe-Register: D1.L Total Size of Required Task-header
D6.W Priority of son-process
OPNAME(A4)=($66.T) 6 Bytes of son's name.
(kann später noch geändert werden).

Ausgabe-Register: A1.L Pointer to generated Task-Header.
CCR EQ if possible, NE if no space av.

Veränderte Register: D1,D7

Funktionsbeschreibung:

Es wird ein Taskkopf der in D1 angegebenen Größe (mind.$62!!) nach Suche von oben nach unten erzeugt. Konnte der benötigte Platz nicht gefunden werden, so retourniert der Trap mit 'NE' und es wird keine Aktion ausgeführt. Anderenfalls antwortet er mit 'EQ' und läd A1 (Zeiger auf den Taskkopf). Der Task-header wird wie folgt vorbesetzt:

```
TYPE   :  User-no. des Aufrufers + TYPE Task
PRIO   :  (Eingangswert D6, nicht mehr änderbar!)
NAME   :  6 bytes aus OPNAME.T des Aufrufers, änderbar.
WSPLEN :  Defaultiert zu $78  (minimaler Platz),  ''
....      No schedule, No buff. activation, no TWS.
BLOCK  :  Waiting for activation (blocked, but linked)
```

Der Anwender muß jetzt unbedingt den Start-PC, $20(A1), auf die zu exekutierende Code-Sequenz bringen! Er kann WSPLEN und NAME noch verändern (etwa bei Langnamen den langrelativen Zeiger einsetzen). Irgendwelche Parameter kann man dem Sohn nur über die Zellen hinter $62(A1) einschreiben, wenn D1.L groß genug war. Der Sohn kann die Parameter dort später mit Hilfe seiner eigenen TID abholen und auswerten.

Nachdem alle Parameter versorgt worden sind, kann man das Blockbyte löschen und einen Dispatcherstart mit dem Trap DPC ($4E43) wagen. Der Sohn setzt sich nun in Gang und raubt - je nach PRIO - eventuell dem erzeugenden Prozeß den Prozessor. Irgenwann wird aber auch der erzeugende Prozeß hinter dem CLR BLOCK(A1) oder dem DPC-Trap fortgesetzt, er muß also entsprechend weitergeführt werden.

Der Trap wird im Bedieninterpreter benutzt, z.B. COPY,LOAD,P,AS, ED,F77, um diesen zeitlich sowie speicherplatzmäßig zu entlasten.

Er ist hervorragend für kompliziertere Shell-extensions geeignet, erfordert allerdings auch ein bißchen Denkarbeit. Wenn der Sohnprozeß nach getaner Arbeit verschwinden soll, so muß man ihn nur auf den Trap TERV ($A010) statt TERMI ($4E41) laufen lassen.

Identify Module By String

I M B S = A 0 3 C

Eingaberegister:	A2.L	Adresse des Modulnamens
Ausgaberegister:	A1.L	Adresse des Moduls
	SR	'EQ' wenn gefunden, sonst 'NE'
Veränderte Register:	D7,A2	

Funktionsbeschreibung:

Der RAM-Bereich des Rechners wird nach einem Modul durchsucht, dessen Name mit dem von A2 adressierten übereinstimmt. Wird das Modul gefunden, antwortet der Trap mit 'EQ' und in A1 steht die Adresse des Moduls. Wird das Modul nicht gefunden, lautet die Antwort 'NE'. Der Name muß mit $FF enden.

Beispiel:

```
IMBS    OPD  $A03C      Trap-Definition
        ....
        LEA  TEXT,A2    Adr. von Modulnamen-> A2
        IMBS            Suchen
        BEQ  FOUND      Springe, wenn gefunden
        ....
TEXT    DC.B 'Mist'     Modulname
        DC.B $FF        Ende des Namens
```

Integer To D1

I N T D 1 = A 0 4 C

Eingaberegister:	A2.L	Adresse des Textes
Ausgaberegister:	D1.L	32 Bit Integer
	SR	'NE' wenn keine Ziffer, sonst 'EQ'
	A2	um Anzahl Ziffern inkr.
Veränderte Register:	D7	

Funktionsbeschreibung:

Eine Zahl in ASCII-Darstellung wird in eine 32 Bit Integer-Zahl gewandelt. In A2 muß die Anfangsadresse der ASCII-Zahl stehen. Das Ergebnis wird in D1 zurückgegeben. Die Umwandlung wird abgebrochen, wenn A2 auf ein ASCII-Zeichen zeigt, welches nicht zwischen 0 ($30) und 9 ($39) zeigt. Liegt gleich das erste Zeichen außerhalb, so antwortet der Trap mit 'NE'. Ist das erste Zeichen eine Ziffer, ist die Antwort 'EQ'. Es wird nicht geprüft, ob ein Overflow auftritt!

Beispiel:

```
INTD1     OPD  $A04C      Trap-Definition
          ....            A2 zeigt auf ASCII-String
          INTD1           Wandeln
          BNE  MÜLL       springe, wenn keine Zahl
          ....
```

I/O-Wait-Function

I O W A = A 0 0 A

Eingaberegister: A1.L
Ausgaberegister: --
Veränderte Register: D7

Funktionsbeschreibung:

Es wird geprüft, ob das mit A1 bezeichnete Element noch in einer Warteschlange steht oder noch in laufender Bearbeitung der Betreuungstask ist. In diesem Fall wird die den 'IOWA' exekutierende Task blockiert im Status 'I/O?'. Ist das Element bereits vollständig bearbeitet, so wirkt der Befehl wie ein 'No Operation'.

Mit der Blockierung der Task wird im CE-Modewort nachträglich das 'Wartebit' gesetzt, so daß die aufrufende Task sofort mit der Beendigung des I/O-Vorganges wieder lauffähig wird.

Die Verwendung dieses Traps empfiehlt sich insbesondere, wenn Inputoperationen früh vor der Benutzung der Daten in Auftrag gegeben wurden und die Daten nun gebraucht werden. Auch nach einem Output ohne Wartebit muß diese Funktion aufgerufen werden, wenn das CE neu parametriert werden soll.

Identify Task By Opname

I T B O = A 0 0 6

Eingaberegister: OPNAME.T Textadresse oder Text
Ausgaberegister: A1.L Adresse der Task
SR 'EQ' wenn gefunden, sonst 'NE'
Veränderte Register: D7,A2

Funktionsbeschreibung:

Der RAM-Bereich des Rechners wird nach einer Task durchsucht. Die Task wird durch den Inhalt von OPNAME.T = $66(A4) beschrieben. In OPNAME.T steht entweder ein 6 Byte langer Text oder eine 4 Byte Adresse eines Textes, der mit $FF enden muß. Wird die Task gefunden, antwortet der Trap mit 'EQ' und in A1 steht die Adresse der Task. Wird die Task nicht gefunden, lautet die Antwort 'NE'.
Beispiel:

```
ITBO      OPD  $A006      Trap-Definition
          ....
          MOVE.L =$41464645,OPNAME.T  'AFFE' nach OPNAME
          MOVE   =$2020,OPNAME+4.T    Mit Blanks auf 6 Byte
          ITBO            Suchen
          BEQ  FOUND      Springe, wenn gefunden
          ....
```

Identify Task By String

I T B S = A 0 4 8

Eingaberegister: A2.L Adresse des Tasknamens
Ausgaberegister: A1.L Adresse der Task
SR 'EQ' wenn gefunden, sonst 'NE'
Veränderte Register: D7,A2

Funktionsbeschreibung:

Der RAM-Bereich des Rechners wird nach einer Task durchsucht, deren Namen mit dem von A2 adressierten übereinstimmt. Wird die Task gefunden, antwortet der Trap mit 'EQ' und in A1 steht die Adresse der Task. Wird die Task nicht gefunden, lautet die Antwort 'NE'. Der Name muß mit $FF enden.

Beispiel:

```
ITBS    OPD  $A048      Trap-Definition
        ....
        LEA  TEXT,A2    Adr. des Tasknamen-> A2
        ITBS            Suchen
        BEQ  FOUND      Springe, wenn gefunden
        ....
TEXT    DC.B 'Mist'     Taskname
        DC.B $FF        Ende des Namens
```

Index-test one-dimension — I T S 1 T = A 0 4 0

Eingaberegister: A1.L Adresse Feldbeschreibungsblock
D0.W linearer Index
Ausgaberegister: -
Veränderte Register: A1,D5,D6,D7

Funktionsbeschreibung:

Es wird geprüft, ob der lineare Index eines eindimensionalen Feldes innerhalb der Feldgrenzen liegt. A1 muß auf den Feldbeschreibungsblock (siehe auch PEARL-Assembler-UP, Kapitel E-IV) zeigen, hier also auf die Feldgrenze der ersten Dimension. Wenn der Index außerhalb der Feldgrenzen liegt, wird die Fehlermeldung 'WRONG INDEX' ausgegeben. War die Zeilenüberwachung eingeschaltet, wird auch die Zeilennummer mit ausgegeben. Der Pearl-Compiler generiert diesen Trap bei der +T-Option.

Beispiel:

```
ITS1T    OPD  $A040     Trap-Definition
         ...            Index in D0.W
         ...            A1.L zeigt auf 1st Dimension
         ITS1T          teste Index
         ...
```

Long Indexx-test one-dimension I T S 1 T L = A 0 6 0

Eingaberegister: A1.L Adresse Feldbeschreibungsblock
D0.L linearer Index
Ausgaberegister: -
Veränderte Register: A1,D5,D6,D7

Funktionsbeschreibung:

Es wird geprüft, ob der lineare Index eines eindimensionalen Feldes innerhalb der Feldgrenzen liegt. A1 muß auf den Feldbeschreibungsblock (siehe auch PEARL-Assembler-UP, Kapitel E-IV) zeigen, hier also auf die Feldgrenze der ersten Dimension. Wenn der Index außerhalb der Feldgrenzen liegt, wird die Fehlermeldung 'WRONG INDEX' ausgegeben. War die Zeilenüberwachung eingeschaltet, wird auch die Zeilennummer mit ausgegeben. Der Pearl-Compiler generiert diesen Trap bei der +T-Option.

Beispiel:

```
ITS1TL   OPD  $A060     Trap-Definition
         ...            Index in D0.L
         ...            A1.L zeigt auf 1st Dimension
         ITS1TL         teste Index
         ...
```

Index-test two-dimension I T S 2 T = A 0 4 2

Eingaberegister: A1.L Adresse Feldbeschreibungsblock
D0.W linearer Index
Ausgaberegister: -
Veränderte Register: A1,D5,D6,D7

Funktionsbeschreibung:

Es wird geprüft, ob der lineare Index eines zweidimensionalen Feldes innerhalb der Feldgrenzen liegt. A1 muß auf den Feldbeschreibungsblock (siehe auch PEARL-Assembler-UP, Kapitel E-IV) zeigen, in dem die Feldgrenzen in umgekehrter Reihenfolge liegen. Wenn der Index außerhalb der Feldgrenzen liegt, wird die Fehlermeldung 'WRONG INDEX' ausgegeben. War die Zeilenüberwachung eingeschaltet, wird auch die Zeilennummer mit ausgegeben. Der Pearl-Compiler generiert diesen Trap bei der +T-Option.

Beispiel:

```
ITS2T    OPD  $A042     Trap-Definition
         ...            Index in D0.W
         ...            A1.L zeigt auf Feldbeschreibungsb.
         ITS2T          teste Index
         ...
```

Long Index-test two-dimension I T S 2 T L = A 0 4 2

Eingaberegister: A1.L Adresse Feldbeschreibungsblock
D0.L linearer Index
Ausgaberegister: -
Veränderte Register: A1,D5,D6,D7

Funktionsbeschreibung:

Es wird geprüft, ob der lineare Index eines zweidimensionalen Feldes innerhalb der Feldgrenzen liegt. A1 muß auf den Feldbeschreibungsblock (siehe auch PEARL-Assembler-UP, Kapitel E-IV) zeigen, in dem die Feldgrenzen in umgekehrter Reihenfolge liegen. Wenn der Index außerhalb der Feldgrenzen liegt, wird die Fehlermeldung 'WRONG INDEX' ausgegeben. War die Zeilenüberwachung eingeschaltet, wird auch die Zeilennummer mit ausgegeben. Der Pearl-Compiler generiert diesen Trap bei der +T-Option.

Beispiel:

```
ITS2TL   OPD  $A062     Trap-Definition
         ...            Index in D0.L
         ...            A1.L zeigt auf Feldbeschreibungsb.
         ITS2TL         teste Index
         ...
```

Index-test three-dimension — I T S 3 T = A 0 4 4

Eingaberegister: A1.L Adresse Feldbeschreibungsblock
D0.W linearer Index
Ausgaberegister: -
Veränderte Register: A1,D5,D6,D7

Funktionsbeschreibung:

Es wird geprüft, ob der lineare Index eines dreidimensionalen Feldes innerhalb der Feldgrenzen liegt. A1 muß auf den Feldbeschreibungsblock (siehe auch PEARL-Assembler-UP, Kapitel E-IV) zeigen, in dem die Feldgrenzen in umgekehrter Reihenfolge liegen. Wenn der Index außerhalb der Feldgrenzen liegt, wird die Fehlermeldung 'WRONG INDEX' ausgegeben. War die Zeilenüberwachung eingeschaltet, wird auch die Zeilennummer mit ausgegeben. Der Pearl-Compiler generiert diesen Trap bei der +T-Option.

Beispiel:

```
ITS3T    OPD  $A044     Trap-Definition
         ...            Index in D0.W
         ...            A1.L zeigt auf Feldbeschreibungsb.
         ITS3T          teste Index
         ...
```

Eingaberegister: A1.L Adresse Feldbeschreibungsblock
D0.L linearer Index
Ausgaberegister: -
Veränderte Register: A1,D5,D6,D7

Funktionsbeschreibung:

Es wird geprüft, ob der lineare Index eines dreidimensionalen Feldes innerhalb der Feldgrenzen liegt. A1 muß auf den Feldbeschreibungsblock (siehe auch PEARL-Assembler-UP, Kapitel E-IV) zeigen, in dem die Feldgrenzen in umgekehrter Reihenfolge liegen. Wenn der Index außerhalb der Feldgrenzen liegt, wird die Fehlermeldung 'WRONG INDEX' ausgegeben. War die Zeilenüberwachung eingeschaltet, wird auch die Zeilennummer mit ausgegeben. Der Pearl-Compiler generiert diesen Trap bei der +T-Option.

Beispiel:

```
ITS3TL    OPD  $A064     Trap-Definition
          ...            Index in D0.L
          ...            A1.L zeigt auf Feldbeschreibungsb.
          ITS3TL         teste Index
          ...
```

Eingaberegister: PC Zeilennummer steht nach PC
BRKADR Break-Adresse
Ausgaberegister: -
Veränderte Register: A1,D1,D6,D7

Funktionsbeschreibung:

Wenn eine Task auf diesen Trap trift und die Zeilennummer in BRKADR = $3E(TID) mit der Zeilennummer nach diesem Trap übereinstimmt, wird sie suspendiert und es erfolgt die Meldung BREAKPOINT SUSPENDED unter Angabe der Zeilennummer. Die Zeilennummer nach dem Trap wird in jedem Fall übersprungen. Desweiteren wird jede überlaufene Zeilennummer in LINENO.T = $A2(A4) eingetragen. Der Pearl-Compiler generiert diesen Trap, wenn die +M-Option eingeschaltet ist und er realen Code erzeugt. Der Eintrag der Zeilennummer nach BRKADR kann mit Hilfe des TRACE-Kommandos erfolgen.

Beispiel:

```
LITRA    OPD  $A036     Trap-Definition
         ...
         LITRA          Line Trace
         DC   $0001     Zeile Nummer 1
         ...            Fortsetzung des Codes
```

Eingaberegister: D1.W Zeilennummer
Ausgaberegister: -
Veränderte Register: A1,D1,D6,D7

Funktionsbeschreibung:

Wenn eine Task auf diesen Trap trift und die Zeilennummer in BRKADR = $3E(TID) mit der Zeilennummer in D1 übereinstimmt, wird sie suspendiert und es erfolgt die Meldung BREAKPOINT SUSPENDED unter Angabe der Zeilennummer. Jede überlaufene Zeilennummer wird in LINENO.T = $A2(A4) eingetragen. Der Pearl-Compiler generiert diesen Trap, wenn die +M-Option eingeschaltet ist und er virtuellen Code erzeugt. Der Eintrag der Zeilennummer nach BRKADR kann mit Hilfe des TRACE-Kommandos erfolgen.

Beispiel:

```
LITRAV    OPD  $A038      Trap-Definition
          ...
          MOVEQ  =1,D1    Zeilennummer nach D1
          LITRAV          Line Trace virtuell
          ...             Fortsetzung des Codes
```

Multiply D2 By 60

M D 2 B 6 0 = A 0 4 6

Eingaberegister: D2.L wird mit 60 multipliziert
Ausgaberegister: D2.L
Veränderte Register: D2,D7

Funktionsbeschreibung:

Der Inhalt von D2 wird mit 60 multipliziert. Das Ergebnis steht ebenfalls in D2. Dieser Trap kann bei der Berechnung der Uhrzeit in Stunden, Minuten, Sekunden verwendet werden.

Beispiel:

```
MD2B60    OPD  $A046      Trap-Definition
          ....            D2 Stunden
          MD2B60          aus Stunden->Minuten
          MOVE.L D2,MIN   Speichern
          MD2B60          aus Min.->Sekunden
          MOVE.L D2,SEC   Speichern
          ....
```

Switch Dispatcher Off

O F F = 4 E 4 F

Eingaberegister: --
Ausgaberegister: --
Veränderte Register: SR

Funktionsbeschreibung:

Der Prozessor wird in den privilegierten Mode der Interruptebene 7 gebracht. Damit kommen keine Interrupts und Dispatchereingriffe mehr zum Zuge.

A c h t u n g !

Dieser Befehl ist mit größter Sorgfalt anzuwenden und darf nur für sehr kurze Zeit (max. ca. 100 µsec) zur Inhibierung der Interrupts führen. Sinn dieser Anweisung ist, bei bestimmten Problemen in E/A-Treibern Sequenzen von wenigen Befehlen unteilbar zu machen.

Der privilegierte Zustand wird durch den Befehl ANDI =$D8FF,SR beendet. Unmittelbar danach muß ein Dispatcherstart forciert werden, z.B. durch Aufruf eines anderen Traps von RTOS-UH oder dem Befehl QDPC (Dispatcherstart=$4E43), sonst können Veränderungen von Taskzuständen 'verschlafen' werden.

```
Eingabe-Register:   D1.L    (Pure usable workspace-size)
                    A5.L    (Is saved in wsp prior to change)
Ausgabe-Register:   A1.L    (address of wsp-section)
                    A5.L    (Pointer to first usable byte)
                    SR      (NE:kein Erfolg, EQ: A1 geladen)
Veränderte Register: D1,D7,A1,A5
```

Funktionsbeschreibung:

Der von RTOS-UH verwaltete Speicherbereich wird von oben nach unten auf die erste freie Sektion durchsucht, in die eine Sektion der in D1 angegebenen Größe samt ihrem Verwaltungskopf hineinpaßt. D1 enthält also die effektiv nutzbare WSP-größe. Wird eine solche freie Sektion nicht gefunden, so antwortet der Trap mit 'NE' anderenfalls mit 'EQ'. Das Register A5 wird in den Verwaltungskopf gerettet und anschließend ebenso wie A1 neu geladen. Register A1 zeigt auf die erzeugte Sektion, A5 auf die Stelle in der Sektion, ab der der Anwender (D1-in) Datenbytes ablegen darf.

Die Umkehroperation hierzu ist der Trap 'RETN' ($4E4C).

Man beachte, daß die so erzeugte Speichersektion als 'PWS', d.h. 'procedure-workspace' verbucht wird und mit der Terminierung der einstmals erzeugenden Task automatisch wieder zu freiem Speicher wird. Dafür sorgt das sog. 'T-link', eine Kette, die ihren Ursprung im 'task-workspace' hat und alle von der Task angeforderten 'CE' (communication element) und 'PWS' miteinander zu einem Ring verbindet. Will man die Sektion von der Task ablösen, wie es zum Beispiel der Editor mit neuen Blöcken macht, so muß die Sektion mit einer besonderen Prozedur aus dem 'T-link' herausgenommen werden. Hierfür empfiehlt sich allerdings nicht dieser Trap sondern der Trap WSBS (A00C).

```
Bsp:        PENTR   OPD      $4E4B
            RETN    OPD      $4E4C
            PRTNAD  EQU      $1A
                    ....
                    BSR     SUBR            Unterprogramm
                    ....
            *-----------------------------------------------*
            SUBR    MOVE.L   =500,D1         500 Bytes Daten
                    PENTR
                    BNE      MIST            B: Kein Platz mehr
                    MOVE.L   (A7)+,PRTNAD(A1)  Return-adr ableg.
                    ....
                    ....     Nun können die Bytes 0(A5) bis incl
                    ....     499(A5) benutzt werden
                    ....
                    RETN                     Wsp zurück+Jump
```

Prevent by name

P R E V = A 0 2 2

Eingaberegister: OPNAME.T Textadresse oder Text
Ausgaberegister: -
Veränderte Register: D7,A1,A2

Funktionsbeschreibung:

Eine Task, deren Name oder deren Adresse des Namens in OPNAME.T = $66(A4) steht, wird ausgeplant. Auch im Aktivierungspuffer aufgelaufene Aktivierungen werden gelöscht. Ist die Task nicht im System vorhanden, wird mit einer entsprechenden Fehlermeldung reagiert und der Aufrufer wird suspendiert.

Beispiel:

```
PREV      OPD  $A022      Trap-Definition
          ....
          MOVE.L =$48414C4C,OPNAME.T  'HALL' nach OPNAME
          MOVE   =$4F20,OPNAME+4.T    'O ' nach OPNAME
          PREV                        Task ausplanen
          ....
```

Prevent quick

P R E V Q = A 0 5 4

Eingaberegister: A1.L Adresse der Task (TID)
Ausgaberegister: -
Veränderte Register: D7,A1

Funktionsbeschreibung:

Eine Task, deren Adresse in A1.L steht, wird ausgeplant. Auch im Aktivierungspuffer aufgelaufene Aktivierungen werden gelöscht.

Beispiel:

```
PREVQ    OPD  $A054     Trap-Definition
         ....           TID in A1
         PREVQ          Task ausplanen
         ....
```

Dispatcher Call

Q D P C = 4 E 4 3

Eingaberegister: -
Ausgaberegister: -
Veränderte Register: D7

Funktionsbeschreibung:

Es wird ein Dispatcherstart forciert. Dieser Trap sollte aufgerufen werden, wenn das Interruptsystem abgeschaltet war (siehe auch OFF). Damit wird verhindert, daß auf Veränderungen von Taskzuständen zu spät reagiert wird. Eine zweite Möglichkeit, den Dispatcher zu starten, ist auch gegeben. Dazu muß vom Dispatcher-Call-Flag (DPC EQU $800) eine Eins subtrahiert werden. Dann wird der Dispatcher bei jedem Trapaufruf oder einem anderen Interrupt gestartet, spätestens also beim nächsten Timer-Interrupt.

Beispiel:

```
     QDPC       OPD  $4E43      Trap-Definition
     DPC        EQU  $800       Dispatcher Call Flag
                ...
                QDPC            Start Dispatcher
                ...

oder:           ...
                SUBQ   =1,DPC   auf 'Minus' setzen
                anderer Trap!   Trap-Aktion+Dispatcher Start
                ...
```

Eingabe-Register: A2.L Adresse des zu untersuchenden Textes.
Ausgabe-Register: A2.L Bei 'EQ' um Stringlänge inkrementiert
SR
Veränderte Register: D7.

Funktionsbeschreibung:

Durch das PC-relative Wort hinter dem Trap wird die Adresse eines Strings übergeben. Der ASCII-string muß mit dem Zeichen $FE beendet sein. QSA prüft nun, ob der String mit der über (A2) erreichbaren Sequenz übereinstimmt. Kleinbuchstaben des (A2)-string werden versuchsweise in Großbuchstaben verwandelt falls der Vergleich eines Zeichens nicht gelingt. Wenn bis zum $FE alle Zeichen des String mit der (A2)-Sequenz übereinstimmen, wird A2 auf das nächste Zeichen der Sequenz vorgerückt und CCR auf 'EQ' gesetzt. Stimmt auch nur ein Zeichen (trotz Kleingroßkonvertierung) nicht überein, so bleibt A2 unverändert und CCR wird auf 'NE' gesetzt.

In jedem Fall werden die beiden auf den Trap folgenden Bytes bei der Rückkehr übersprungen.

Der Trap eignet sich für eine einfache Textanalyse und wird innerhalb des Bedieninterpreters eingesetzt. Er ist darum auch zur Realisierung neuer Bedienbefehle optimal geeignet.

BEISPIEL:

```
QSA     OPD    $A01E       Trap-definition
        .....              A2 zeigt auf Eingabetext
        QSA                Aufruf
        DC     TEXT-$      Stringadresse relativiert.
        BEQ    Irgendwo    Springt bei erhöhtem A2 nach Irgendwo.
        ...
TEXT    DC.B   'STIMMTS',$FE
```

Der BEQ wird ausgeführt wenn A2 vor dem TRAP z.B. auf einen Text der Form Stimmtsblabla oder sTIMMtSxx etc. zeigt. Nach dem QSA zeigt A2 dann z.B. auf 'blabla' bzw. 'xx'.

Read-Clock

R C L K = A 0 3 E

Eingaberegister: -
Ausgaberegister: D1.L Uhrzeit in Millisekunden
Veränderte Register: D7

Funktionsbeschreibung:

Die aktuelle Uhrzeit wird gelesen und in D1 zurückgegeben. D1 enthält die Uhrzeit in Millisekunden.

Beispiel:

```
RCLK    OPD  $A03E      Trap-Definition
        ....
        RCLK            Uhrzeit lesen
        MOVE.L D1,TEMP  Uhrzeit speichern
        ....
```

Eingaberegister: A1.L
Ausgaberegister: --
Veränderte Register: D5,D6,D7,A1

Funktionsbeschreibung:

An Hand der Verwaltungszeiger des in A1.L bezeichneten CEs wird zunächst geprüft, ob das CE zur Zeit in einen E/A-Vorgang verwickelt ist. Ist das nicht der Fall, wird es sofort in freien Speicher umgewandelt.

Wenn dagegen das CE in einer Warteschlange steht und noch nicht in Bearbeitung einer Betreuungstask ist, so wird lediglich das Release-Bit gesetzt.

Im Fall, daß das CE bereits in Bearbeitung durch eine Betreuungstask ist, müssen zwei Fälle unterschieden werden:

Fall a) Die exekutierende Task ist Besitzer des CEs.
In diesem Fall wird nur das Release-Bit in STATIO gesetzt.

Fall b) Die exekutierende Task ist nicht Besitzer des CEs.
Es wird angenommen, daß die aufrufende Task eine Betreuungstask ist. Falls bei TIDO kein Besitzer mehr eingetragen ist, wird das CE sofort zu freiem Speicherraum. Ist aber ein Besitzer eingetragen, so wird das Release-Bit in STATIO inspiziert: ist es gesetzt, wird das CE zu freiem Speicher, wenn nicht, werden nur die Verwaltungszeiger so verändert, daß erkennbar ist (z.B. beim IOWA), daß dieses CE nicht in eine I/O-Operation verwickelt ist.

Die Benutzung dieses Befehls mit einer Besetzung von A1, die nicht auf ein existierendes CE zeigt, wird wahrscheinlich zu einem Systemzusammenbruch führen!

Eingabe-Register: A1.L Adresse der (2 byte) Semavariablen.
Ausgabe-Register: --
Veränderte Register: D5,D6,D7

Funktionsbeschreibung:

Die angegebene Semavariable (per Adresse in A1) wird um 1 erhöht. Ist der neue Wert größer als Null, so erfolgt unmittelbar ohne weitere Aktion die Rückkehr aus dem Trap. Wird nach dem Erhöhen jedoch der Wert Null angenommen, so wird ihr Wert auf 1 gesetzt. Alle durch 'SEMA' blockierten Tasks werden untersucht, ob sie auf diese Veränderung hin wieder lauffähig werden. Wird keine Task gefunden, erfolgt gleichfalls Rückkehr (mit dem Wert 1) ohne weitere Aktion. Wurde jedoch eine Task gefunden -bei mehreren gilt das für die höchstpriorisierte- , so wird dieser die Semaphore zugeteilt und der Wert wieder auf -1 ('Task waiting') gesetzt. Die Task wird nach der Zuteilung wieder lauffähig.

Man beachte, daß der 'Außenwelt' gegenüber der zeitweilig angenommene Wert -1 gleichwertig zur Null (Requested) ist und nur interne Bedeutung hat (Requested and task waiting).

```
Bsp:        RELEA     OPD     $4E47

                      ....

                      LEA     >SEMA4,A1       Globale Semavariable
                      RELEA

                      ....
```

Request semaphore. R E Q U = 4 E 4 6

Eingabe-Register: A1.L Adresse der (2 byte) Semavariablen.
Ausgabe-Register: --
Veränderte Register: D7, im Task-WSP: OPFATI ($6C im TWS).

Funktionsbeschreibung:

Die angegebene Semavariable (per Adresse in A1) wird um 1 erniedrigt. Ist der neue Wert größer oder gleich Null, so erfolgt die Rückkehr ohne weitere Aktion. Ein negativer Wert jedoch wird auf -1 korrigiert und die anfordernde Task wird blockiert. Im Taskworkspace der anfordernden Task (OPFATI=$6C+TWS-adr) wird eine Notiz hinterlassen, die später ggf. vom Trap RELEA aufgefunden werden kann. Wie bei allen Traps, so ist auch hier ein Dispatchereingriff innerhalb des Trap-body nicht möglich, so daß die notwendige Unteilbarkeit des Request automatisch durch die Struktur von RTOS-UH gesichert ist. Obwohl der Taskworkspace benutzt wird, braucht A4 nicht besetzt zu sein. Der Trap holt sich die entsprechende Adresse des TWS auf andere Weise.

Man beachte, daß der 'Außenwelt' gegenüber der zeitweilig angenommene Wert -1 gleichwertig zur Null (Requested) ist und nur interne Bedeutung hat (Requested and task waiting).

```
Bsp:        REQU     OPD     $4E46

                     ....

                     LEA     >SEMA4,A1      Globale Semavariable
                     REQU                   Je nach vorherigem
                                            Zustand wird die
                                            Task evtl. blockiert

                     ....
```

Rubber Blanks

R U B B L = A 0 2 0

Eingaberegister: A2.L Adresse des zu unters. Textes
Ausgaberegister: A2.L Um Anzahl Blanks inkrem.
Veränderte Register: D7

Funktionsbeschreibung:

In einem Text können Leerzeichen überlesen werden. A2 zeigt auf den Text und wird solange inkrementiert, wie Leerzeichen hintereinander im Text stehen. Dieser Trap kann bei der Realisierung von neuen Bedienbefehlen eingesetzt werden.

Beispiel:

```
RUBBL    OPD  $A020     Trap-Definition
         ....           A2 zeigt auf Text
         RUBBL          Überliest Blanks
         ....           Weitere Textanalyse
```

Return from procedure. R E T N = 4 E 4 C

Eingabe-Register: A5.L Zeiger auf zuletzt einger. workspace
Ausgabe-Register: A5.L Wird aus dem freigebenen WSP geladen.
PC Wird aus dem '' '' ''
Veränderte Register: A1,D7

Funktionsbeschreibung:

Der früher mit PENTR eingerichtete PWSP (procedure-workspace) wird mit diesem Trap wieder freigegeben. Vorher wird allerdings das vom PENTR gerette Register A5 zurückgeladen. Der Trap kehrt nicht zum Aufrufer zurück, sondern holt sich die Rückkehradresse ebenfalls aus dem Verwaltungskopf der mit A5(in) bezeichneten Speichersektion. Auf PRTNAD, erreichbar über -8(A5), muß also unbedingt eine verwertbare Fortsetzungsadresse stehen. (Der Trap PENTR schreibt dort nichts hin!).

Hinweis:

Der Trap eignet sich zur Rückkehr aus PEARL-Prozeduren, wenn diese mit dem Hyperprocbefehl 'ENTR' beginnen, den ENTR erledigt neben der PWS-Erzeugung mit PENTR auch die Ablage der Rückkehradresse.

Bsp.

```
PENTR   OPD      $4E48
RETN    OPD      $4E4C
PRTNAD  EQU      $1A              $1A(A1) oder -8(A5)
        ....
        BSR     SUBR              Unterprogramm
        ....
*-------------------------------------------------------*
SUBR    MOVE.L   =500,D1          500 Bytes Daten
        PENTR
        BNE      MIST             B: Kein Platz mehr
        MOVE.L   (A7)+,PRTNAD(A1)  Return-adr ableg.
        ....
        ....     Nun können die Bytes 0(A5) bis incl
        ....     499(A5) benutzt werden
        ....
        RETN                      Wsp zurück+Jump
```

Rubber Blanks

R U B B L = A 0 2 0

Eingaberegister: A2.L Adresse des zu unters. Textes
Ausgaberegister: A2.L Um Anzahl Blanks inkrem.
Veränderte Register: D7

Funktionsbeschreibung:

In einem Text können Leerzeichen überlesen werden. A2 zeigt auf den Text und wird solange inkrementiert, wie Leerzeichen hintereinander im Text stehen. Dieser Trap kann bei der Realisierung von neuen Bedienbefehlen eingesetzt werden.

Beispiel:

```
RUBBL    OPD  $A020     Trap-Definition
         ....           A2 zeigt auf Text
         RUBBL          Überliest Blanks
         ....           Weitere Textanalyse
```

Eingabe-Register: A1.L Adresse der zu löschenden Sektion.
Ausgabe-Register: --
Veränderte Register: D7.

Funktionsbeschreibung:

Die angegebene Speichersektion wird in freien Speicherraum zurückverwandelt. Anschließend darf A1 nicht mehr als Zeiger verwendet werden. Falls irgendwo Tasks auf die Zuteilung von 'TWS' (Task-workspace) oder 'PWS' (Prozedur-workspace) gewartet hatten, erfolgt nunmehr möglicherweise ihre Freigabe - falls der Platzbedarf nun befriedigt werden kann.

Wichtiger Hinweis:

Mit diesem Trap dürfen nur solche Sektionen gelöscht werden, die weder im 'S'- (Dispatcher, I/O-queue) noch im 'T'-link (Task-link) eingekettet sind, schwere später auftretende Abstürze können sonst resultieren. So müssen z.B. mit WSFA,WSBS und WSFS eingerichtete 'PWS'-Sektionen ausgelinkt werden, bevor dieser Trap eingesetzt werden darf. Das Auslinken ist bei WSFS etc. beschrieben. Mit PENTR eingerichtete Sektionen werden dagegen nicht mit RWSP rückgegeben, sondern durch RETN (ohne vorheriges Auslinken!) gelöscht. Ebenso werden mit FETCE erzeugte 'CE' (Communication-elemente) mit RELCE (auch ohne vorheriges Auslinken) getilgt.

Der Trap RWSP wird intern in den Traps RELCE und RETN benutzt, nachdem diese das Auslinken, sowie ihre Nebenoperationen erledigt haben. Auch innerhalb des UNLOAD-Befehles sowie im Trap TERV wird er benutzt. Der Editor verwendet RWSP beim 'ERASE'.

```
Bsp:        RWSP     OPD     $A02A
            WSBS     OPD     $A00C
                     ....

                     MOVE.L  ...,D1        Wsp-Size
                     WSBS                  Wsp anfordern
                     Auslinken!            siehe WSBS-Beschr.
                     ....                  Nun als 'Dauerblock'
                     ....                  benutzbar, evtl auch
                     ....                  in anderen Tasks.

                     RWSP                  Sektion zurückgeben.
```

Eingaberegister:	D7.L	Nummer der Scheibe
	D6.L	ggf. bei Fortsetzung der Suche
Ausgaberegister:	A1.L	Adresse der Scheibe
	SR	'EQ' wenn gefunden, sonst 'NE'
Veränderte Register:	D5,D6,D7	

Funktionsbeschreibung:

Es wird der Scan-Bereich, der beim Nukleus eingetragen ist (siehe RTOS-UH Autolink/Seite E-VII), nach der Scheibe, deren Nummer in D7 übergeben wird, durchsucht. Wenn die Scheibe gefunden wird, antwortet der Trap mit 'EQ' und man erhält in A1 die Adresse nach der Signalmarke. Wird die Scheibe nicht gefunden, antwortet der Trap mit 'NE'. Wenn D7.L und D6.L nicht verändert werden, kann der Trap zur Fortsetzung der Suche wieder aufgerufen werden.

Beispiel:

```
SCAN     OPD   $4E45      Trap-Definition
         ...
         MOVEQ  =1,D7     Scheibe 1
LABEL    SCAN             suchen
         BNE.S  READY     B: nichts zu finden
         ...              Action ...
         ...              aber D6,D7 bleiben unverändert
         BRA.S  LABEL     Fortsetzung der Suche
READY    ...              hier geht's weiter...
```

Set Battery Clock

S T B C L K = A 0 5 E

Eingaberegister: -
Ausgaberegister: D2.L Zeit in Millisekunden
Veränderte Register: D7

Funktionsbeschreibung:

Dieser Trap wird beim CLOCKSET-Kommando exekutiert. Normalerweise wird damit keine Aktion ausgeführt. Er kann aber vom Nutzer angeschlossen werden, um eine eventuell vorhandene Hardware-Uhr zu stellen. In D2.L erhält man die Zeit, die beim CLOCKSET-Kommando eingegeben wurde, in Millisekunden übergeben. Der Anschluß des Trap kann von RTOS in der Startphase über die entsprechende Scheibe 14 gemacht werden.

Suspend actual running Task

S U S P = A 0 2 8

Eingaberegister: -
Ausgaberegister: -
Veränderte Register: D7,A1

Funktionsbeschreibung:

Die gerade laufende Task wird suspendiert. Alle Register werden gerettet. Die Task behält ihren Workspace und kann zu einem späteren Zeitpunkt fortgesetzt werden.

Beispiel:

```
SUSP    OPD  $A028    Trap-Definition
        ....          Task läuft
        SUSP          Task suspendiert sich
        ....
```

Eingaberegister: OPNAME.T Textadresse oder Text
Ausgaberegister: -
Veränderte Register: D1,D5,D6,D7,A1,A2

Funktionsbeschreibung:

Die Task, deren Name oder deren Zeiger auf den Namen in OPNAME.T = $66(A4) steht wird sofort beendet. Die Registerinhalte werden nicht gerettet, gehen also verloren. Bei einer nichtresidenten Task werden alle Speichersektionen (PWS, TWS, CE) an das Betriebssystem zurückgegeben. Residente Tasks behalten ihren Taskworkspace. Die Ausgabe-CEs verbleiben in der Schlange, während Eingabe-CEs, die nicht schon von der E/A-Betreuungstask bearbeitet werden, aus der Schlange entfernt werden.

Beispiel:

```
TERME     OPD  $A000      Trap-Definition
          ....
          MOVE.L =$57454720,OPNAME.T  'WEG '
          MOVE   =$2020,OPNAME+4.T    Auffüllen auf 6 Byte
          TERME          Task WEG terminieren
          ....
```

Terminate quick

T E R M E Q = A 0 5 8

Eingaberegister: A1.L Adresse der Task (TID)
Ausgaberegister: --
Veränderte Register: D1,D5,D6,D7,A1

Funktionsbeschreibung:

Die Task, deren Adresse in A1.L steht, wird sofort beendet. Die Registerinhalte werden nicht gerettet und gehen somit in jedem Fall verloren. Ist die Task vom nichtresidenten Typ, so werden alle Speichersektionen (PWS, TWS, CE) an RTOS-UH zurückgegeben. Residente Tasks behalten dagegen ihren Taskworkspace. Communication-Elemente in der Ausgabeschlange verbleiben dort, aus der Eingabeschlange werden sie dagegen herausgelöst, es sei denn, daß sie bereits von der E/A-Betreuungstask bearbeitet werden.

Eingeplante Tasks verbleiben auch nach dieser Anweisung in der Dispatcherverwaltung. War bereits eine neue Aktivierung gepuffert, so wird die Task der gepufferten Aktivierung entsprechend mit ggf. anderer Priorität erneut lauffähig gemacht.

Beispiel:

```
TERMEQ    EQU  $A058     Trap-Definition
          ...            TID nach A1.L
          TERMEQ         Terminate Task
          ...
```

Eingaberegister: --
Ausgaberegister: --
Veränderte Register: alle

Funktionsbeschreibung:

Eine Task, die diesen Befehl exekutiert, wird sofort beendet. Die Registerinhalte werden nicht gerettet und gehen somit in jedem Fall verloren. Ist die Task vom nichtresidenten Typ, so werden alle Speichersektionen (PWS, TWS, CE) an RTOS-UH zurückgegeben. Residente Tasks behalten dagegen ihren Taskworkspace. Communication-Elemente in der Ausgabeschlange verbleiben dort, aus der Eingabeschlange werden sie dagegen herausgelöst, es sei denn, daß sie bereits von der E/A-Betreuungstask bearbeitet werden.

Eingeplante Tasks verbleiben auch nach dieser Anweisung in der Dispatcherverwaltung. War bereits eine neue Aktivierung gepuffert, so wird die Task der gepufferten Aktivierung entsprechend mit ggf. anderer Priorität erneut lauffähig gemacht.

Terminate and Vanish

T E R V = A 0 1 0

Eingaberegister: -
Ausgaberegister: -
Veränderte Register: alle

Funktionsbeschreibung:

Es werden die gleichen Aktionen ausgeführt wie beim TERMI ($4E41). Zusätzlich verschwindet die Task aus dem System. Sie kann nicht wieder aktiviert werden, außer sie war vorher eingeplant. Dann bleibt sie solange im System, bis die Einplanung erlischt.

Dieser Trap kann dazu genutzt werden, um eine Subtask, die mit GAPST erzeugt wurde, mit Erledigung ihrer Aufgabe aus dem System zu entfernen.

Beispiel:

```
TERV    OPD  $A010     Trap-Definition
        ....           Subtask
        ....
        TERV           Ende der Subtask
```

Eingaberegister:	D1.W	Priorität
	OPNAME.T	Textadresse oder Text
	OPFATI.T	Startzeit
	OPINTV.T	Intervall
	OPLTI.T	Endzeit
Ausgaberegister:	-	
Veränderte Register:	D1,D6,D7,A1,A2	

Funktionsbeschreibung:

Eine Task, deren Name oder deren Adresse des Namens in OPNAME.T = $66(A4) steht, wird zur Aktivierung eingeplant. In D1 wird die Priorität der Aktivierung übergeben. Ist D1 gelöscht, wird die Default-Priorität eingetragen. Die aktuelle Priorität der laufenden Task bleibt unbeeinflußt. Wenn die Task nach einer bestimmten Zeitdauer aktiviert werden soll, muß in OPFATI.T = $6C(A4) die Zeitdauer in Millisekunden mit gesetztem obersten Bit eingetragen sein. Soll die Aktivierung zu einem bestimmten Zeitpunkt erfolgen, darf das oberste Bit nicht gesetzt sein. Ist eine wiederholte Einplanung gewünscht, muß in OPINTV.T = $36(A4) der Abstand der Einplanungen eingetragen werden. Ist der Wert negativ oder Null, findet keine zyklische Einplanung statt. Sollen die Einplanungen zu einem bestimmten Zeitpunkt oder nach einer Zeitdauer (oberstes Bit gesetzt) beendet werden, muß OPLTI.T = $3A(A4) mit dem entsprechenden Wert geladen werden. Bestehende Einplanungen werden gelöscht. Die Fehlermeldungen entsprechen denen beim Trap ACT beschriebenen.

Beispiel:

```
TIAC    OPD  $A016            Trap-Definition
        ...                   TID in A1
        MOVE.L =$80002710,OPFATI.T  after 10 sec
        MOVE.L =1000,OPINTV.T       all 1 sec
        CLR.L  OPLTI.L        ohne Ende
        TIAC                  nach 10 Sec aktivieren
        ...
```

Eingaberegister:	D1.W	Priorität
	A1.L	Adresse der Task (TID)
	OPFATI.T	Startzeit
	OPINTV.T	Intervall
	OPLTI.T	Endzeit
Ausgaberegister:	-	
Veränderte Register:	D1,D6,D7,A1,A2	

Funktionsbeschreibung:

Eine Task, deren Adresse in A1 steht, wird zur Aktivierung eingeplant. In D1 wird die Priorität der Aktivierung übergeben. Ist D1 gelöscht, wird die Default-Priorität eingetragen. Die aktuelle Priorität der laufenden Task bleibt unbeeinflußt. Wenn die Task nach einer bestimmten Zeitdauer aktiviert werden soll, muß in OPFATI.T = $6C(A4) die Zeitdauer in Millisekunden mit gesetztem obersten Bit eingetragen sein. Soll die Aktivierung zu einem bestimmten Zeitpunkt erfolgen, darf das oberste Bit nicht gesetzt sein. Ist eine wiederholte Einplanung gewünscht, muß in OPINTV.T = $36(A4) der Abstand der Einplanungen eingetragen werden. Ist der Wert negativ oder Null, findet keine zyklische Einplanung statt. Sollen die Einplanungen zu einem bestimmten Zeitpunkt oder nach einer Zeitdauer (oberstes Bit gesetzt) beendet werden, muß OPLTI.T = $3A(A4) mit dem entsprechenden Wert geladen werden. Bestehende Einplanungen werden gelöscht.

Beispiel:

```
TIACQ     OPD  $A024            Trap-Definition
          ...                   TID in A1
          MOVE.L =$80002710,OPFATI.T  after 10 sec
          MOVE.L =1000,OPINTV.T       all 1 sec
          CLR.L  OPLTI.L        ohne Ende
          TIACQ                 nach 10 Sec aktivieren
          ...
```

Eingaberegister: OPNAME.T Textadresse oder Text
OPFATI.T Zeitdauer oder Zeitpunkt
Ausgaberegister: -
Veränderte Register: D6,D7,A1,A2

Funktionsbeschreibung:

Eine Task, deren Name oder deren Adresse des Namens in OPNAME.T = $66(A4) steht, wird zur Fortsetzung eingeplant. Bestehende Einplanungen werden gelöscht. Wenn die Task nach einer bestimmten Zeitdauer fortgesetzt werden soll, muß in OPFATI.T = $6C(A4) die Zeitdauer in Millisekunden mit gesetztem obersten Bit eingetragen sein. Soll die Fortsetzung zu einem bestimmten Zeitpunkt erfolgen, darf das oberste Bit nicht gesetzt sein.

Beispiel:

```
TICON   OPD  $A018            Trap-Definition
        ...
        LEA     TSKNAM,A0     Adresse des Tasknamens
        MOVE.L A0,OPNAME.T    Eintrag der Adresse
        CLR     OPNAME+4.T    kein Text
        MOVE.L =$800003E8,OPFATI.T  1 sec Resume
        TICON                 nach 1 Sec fortsetzen
        ...
```

Time continue quick

T I C O N Q = A 0 4 E

Eingaberegister: A1.L Adresse der Task (TID)
OPFATI.T Zeitdauer oder Zeitpunkt
Ausgaberegister: -
Veränderte Register: D6,D7,A1,A2

Funktionsbeschreibung:

Eine Task, deren Adresse in A1 steht, wird zur Fortsetzung eingeplant. Bestehende Einplanungen werden gelöscht. Wenn die Task nach einer bestimmten Zeitdauer fortgesetzt werden soll, muß in OPFATI.T = $6C(A4) die Zeitdauer in Millisekunden mit gesetztem obersten Bit eingetragen sein. Soll die Fortsetzung zu einem bestimmten Zeitpunkt erfolgen, darf das oberste Bit nicht gesetzt sein.

Beispiel:

```
TICONQ    OPD  $A04E        Trap-Definition
          ...               TID in A1
          MOVE.L =$800003E8,OPFATI.T  1 sec Resume
          TICONQ            nach 1 Sec fortsetzen
          ...
```

Time Resume

T I R E = A 0 2 C

Eingaberegister: OPFATI.T Zeitdauer oder Zeitpunkt
Ausgaberegister: -
Veränderte Register: D6,D7,A1

Funktionsbeschreibung:

Die Task, die diesen Trap absetzt, wird suspendiert und nach der angegebenen Zeit oder zu einem bestimmten Zeitpunkt fortgesetzt. Bestehende Einplanungen werden gelöscht. Wenn die Task nach einer bestimmten Zeitdauer fortgesetzt werden soll, muß in OPFATI.T = $6C(A4) die Zeitdauer in Millisekunden mit gesetztem obersten Bit stehen. Andernfalls wird die eingetragene Zeit als Zeitpunkt interpretiert und zu diesem wird die Task zur Fortsetzung eingeplant.

Beispiel:

```
TIRE    OPD  $A02C        Trap Definition
        ...
        MOVE.L  =$80000008,OPFATI.T  8 msec Resume
        TIRE            nach 8 msec fortsetzen
        ...
```

Take of Queue

T O Q = 4 E 4 D

Eingaberegister: D1.W
Ausgaberegister: A1.L,PC
Veränderte Register: A1,D7

Funktionsbeschreibung:

In D1.W wird die Nummer einer Betreuungstask erwartet. Ist die Warteschlange leer, so wird die Exekution mit dem nächsten folgenden Befehl fortgesetzt. Ist die Warteschlange nicht leer, so wird A1 mit dem Zeiger auf das vorne - am Kopf der Schlange - stehende Communication-Element geladen. Vor der Rückkehr wird jetzt der PC um 2 erhöht, so daß der nächstfolgende Einwortbefehl übersprungen wird. Das Element ist danach ausgekettet und sein Vorwärtszeiger 'S' negativ, um die in 'RELCE' und 'IOWA' enthaltenen Abfragen korrekt vorzubereiten.

Es wird **nicht** geprüft, ob D1 im implementierten Bereich liegt! Bei der typischen Anwendung dieser Funktion in einer I/O-Betreuungstask wird D1 ohnehin 'immidiate' mit einer Konstanten geladen. Die Systemreaktion bei einer falschen Parametrierung von D1 ist nicht vorherzusagen, kann u.U. aber zur Katastrophe führen.

Beispiel:

```
TOQ   EQU    $4E40
TERMI EQU    $4E41
...
      MOVE   =0,D1     LDN A1
      TOQ              CE aus Schlange holen
      TERMI            Wenn Schlange leer TERMI
...                    Verarbeitung des CEs
```

Eingaberegister: D1.L Interrupt-Maske
Ausgaberegister: -
Veränderte Register: D7

Funktionsbeschreibung:

Alle Interrupts, die durch ein gesetztes Bit in D1.L beschrieben werden und die 'enabled' sind, werden gefeuert. Damit werden alle Tasks, die auf die entsprechenden Interrupts zur Aktivierung oder Fortsetzung eingeplant sind, freigegeben. Im System wird nicht unterschieden, ob ein Interrupt von der Hardware oder dem TRIGEV-Trap ausgelöst wurde. Somit ist es also möglich, Hardware-Interrupts zu simulieren und ihre Wirkung zu testen.

Beispiel:

```
TRIGEV    OPD  $A026      Trap-Definition
          ...
          MOVEQ  =1,D1    EV 1
          TRIGEV          trigger EV 1
          ...
```

Eingabe-Register: D1.L (Total size of requ Workspace)
Ausgabe-Register: A1.L (Address of memory-section)
SR (NE:kein Erfolg, EQ: A1 geladen)
Veränderte Register: D1,D7

Funktionsbeschreibung:

Der von RTOS-UH verwaltete Speicherbereich wird von oben nach unten nach dem ersten freien Stück abgesucht, in welches die Anforderung hineinpaßt. Der untere Rest davon bleibt freie Sektion. A1 wird so geladen, daß es auf den oben bündig liegenden Bereich zeigt. Das Statusregister SR wird zur Rückantwort benutzt. Wenn nämlich kein Platz gefunden wurde, so antwortet der Trap mit 'NE' anderenfalls (wenn A1 geladen werden konnte) mit 'EQ'.

Man beachte, daß die so erzeugte Speichersektion als 'PWS', d.h. 'procedure-workspace' verbucht wird und mit der Terminierung der einstmals erzeugenden Task automatisch wieder zu freiem Speicher wird. Dafür sorgt das sog. 'T-link', eine Kette, die ihren Ursprung im 'task-workspace' hat und alle von der Task angeforderten 'CE' (communication element) und 'PWS' miteinander zu einem Ring verbindet. Will man die Sektion von der Task ablösen, wie es zum Beispiel der Editor mit neuen Blöcken macht, so muß die Sektion mit einer besonderen Prozedur aus dem 'T-link' herausgenommen werden, die unten erläutert ist.

```
WSBS    OPD       $A00C
        MOVE.L    =$1000,D1       Sektion 4 kB brutto
        WSBS
        BNE       MIST            B: kein Platz mehr
--------- Nur wenn 'Dauerblock' gewünscht wird: ----------
        OFF                       unteilbare Sequenz
        MOVEA.L   BACKT(A1),A2          A2 zuf. Beisp.
        MOVE.L    FORT(A1),FORT(A2)     linker  Nachb.
        MOVEA.L   FORT(A1),A2
        MOVE.L    BACKT(A1),BACKT(A2)   rechter Nachb.
        MOVE      =$0010,TYPE(A1) als 'MODULE' ausg.
        MOVE.... text6-> NAME(A1)  Modulname 6 Bytes
        ANDI      =$D8FF,SR       Sperre löschen
        DPC  .. falls nicht bald anderer Trap kommt!
```

Die symbolischen Displacements: BACKT=14, FORT=NAME=10,TYPE=8.
Ein so erzeugter Dauerbereich kann nur über UNLOAD name wieder eliminiert werden. Ab Displacement 22 können Daten abgelegt werden. (Bei ausgelinkten Sektionen notfalls ab Displacement 16)

Der Trap WSBS ist ein Bruder der Traps WSFA, WSFS und PENTR, die ähnliche Grundeigenschaften aufweisen.

Gewisse Sorgfalt bei der Anwendung versteht sich von selbst!

Workspace fixed address request — W S F A = A 0 0 8

```
Eingabe-Register:    D1.L    (Requested address, must be even)
                     A1.L    (Adr of first unused word,  ''  )
Ausgabe-Register:    A1.L    (Sect. address, A1 out = D1 in)
                     SR      (NE:kein Erfolg, EQ: A1 geladen)
Veränderte Register: D1,D5,D6,D7
```

Funktionsbeschreibung:

Der Trap prüft, ob zwischen der Untergrenze in D1.L und der Obergrenze+2 in A1.L freier Bereich liegt. Ist dies nicht der Fall, so antwortet der Trap mit 'NE', anderenfalls mit 'EQ'. Eventuelle Reste oberhalb und unterhalb werden - falls groß genug - zu freien Sektionen. Der Ausgangswert von A1 ist identisch mit dem Eingangswert von D1 (falls 'EQ').

Man beachte, daß die so erzeugte Speichersektion als 'PWS', d.h. 'procedure-workspace' verbucht wird und mit der Terminierung der einstmals erzeugenden Task automatisch wieder zu freiem Speicher wird. Dafür sorgt das sog. 'T-link', eine Kette, die ihren Ursprung im 'task-workspace' hat und alle von der Task angeforderten 'CE' (communication element) und 'PWS' miteinander zu einem Ring verbindet. Will man die Sektion von der Task ablösen, wie es zum Beispiel der Editor mit neuen Blöcken macht, so muß die Sektion mit einer besonderen Prozedur aus dem 'T-link' herausgenommen werden, die unten erläutert ist.

```
WSFA    OPD       $A008
        MOVE.L    =$8000,D1        Sektionadr=$8000
        LEA       $9000.L,A1       Bereich:$8000-$8FFF
        WSFA
        BNE       MIST             B:Bereich nicht frei
--------- Nur wenn 'Dauerblock' gewünscht wird: ----------
        OFF                          unteilbare Sequenz
        MOVEA.L   BACKT(A1),A2           A2 zuf. Beisp.
        MOVE.L    FORT(A1),FORT(A2)      linker  Nachb.
        MOVEA.L   FORT(A1),A2
        MOVE.L    BACKT(A1),BACKT(A2)  rechter Nachb.
        MOVE      =$0010,TYPE(A1) als 'MODULE' ausg.
        MOVE.... text6-> NAME(A1)  Modulname 6 Bytes
        ANDI      =$D8FF,SR        Sperre löschen
        DPC  .. falls nicht bald anderer Trap kommt!
```

Die symbolischen Displacements: BACKT=14, FORT=NAME=10,TYPE=8. Ein so erzeugter Dauerbereich kann nur über UNLOAD name wieder eliminiert werden. Ab Displacement 22 können Daten abgelegt werden. (Bei ausgelinkten Sektionen notfalls ab Displacement 16)

Der Trap WSFA ist ein Bruder der Traps WSBS, WSFS und PENTR, die ähnliche Grundeigenschaften aufweisen.

Natürlich stört dieser Trap die automatische Speicherverwaltung.

Eingabe-Register: D1.L (Total size of requ Workspace)
Ausgabe-Register: A1.L (Address of memory-section)
SR (NE:kein Erfolg, EQ: A1 geladen)
Veränderte Register: D1,D7

Funktionsbeschreibung:

Der von RTOS-UH verwaltete Speicherbereich wird von unten nach oben nach dem ersten freien Stück abgesucht, in welches die Anforderung hineinpaßt. Dann wird A1 als Zeiger auf diesen Abschnitt geladen. Der ggf. vorhandene (obere) Rest wird zur freien Sektion verwandelt. Das Statusregister wird zur Rückantwort benutzt. Wenn nämlich kein Platz gefunden wurde, so antwortet der Trap mit 'NE' anderenfalls (wenn A1 geladen werden konnte) mit 'EQ'.

Man beachte, daß die so erzeugte Speichersektion als 'PWS', d.h. 'procedure-workspace' verbucht wird und mit der Terminierung der einstmals erzeugenden Task automatisch wieder zu freiem Speicher wird. Dafür sorgt das sog. 'T-link', eine Kette, die ihren Ursprung im 'task-workspace' hat und alle von der Task angeforderten 'CE' (communication element) und 'PWS' miteinander zu einem Ring verbindet. Will man die Sektion von der Task ablösen, wie es zum Beispiel der Editor mit neuen Blöcken macht, so muß die Sektion mit einer besonderen Prozedur aus dem 'T-link' herausgenommen werden, die unten erläutert ist.

```
                MOVE.L   =$1000,D1      Sektion 4 kB brutto
                WSFS
                BNE      MIST           B: kein Platz mehr
--------- Nur wenn 'Dauerblock' gewünscht wird: ----------
                OFF                      unteilbare Sequenz
                MOVEA.L  BACKT(A1),A2         A2 zuf. Beisp.
                MOVE.L   FORT(A1),FORT(A2)    linker  Nachb.
                MOVEA.L  FORT(A1),A2
                MOVE.L   BACKT(A1),BACKT(A2)  rechter Nachb.
                MOVE     =$0010,TYPE(A1) als 'MODULE' ausg.
                MOVE.... text6-> NAME(A1)  Modulname 6 Bytes
                ANDI     =$D8FF,SR       Sperre löschen
                DPC  .. falls nicht bald anderer Trap kommt!
```

Die symbolischen Displacements: BACKT=14, FORT=NAME=10,TYPE=8.
Ein so erzeugter Dauerbereich kann nur über UNLOAD name wieder eliminiert werden. Ab Displacement 22 können Daten abgelegt werden. (Bei ausgelinkten Sektionen notfalls ab Displacement 16)

Der Trap WSFS ist ein Bruder der Traps WSFA, WSBS und PENTR, die ähnliche Grundeigenschaften aufweisen.

Gewisse Sorgfalt bei der Anwendung versteht sich von selbst!

Transfer CE for Input/Output — X I O = 4 E 4 A

Eingaberegister: A1.L
Ausgaberegister: --
Veränderte Register: D1,D5,D6,D7

Funktionsbeschreibung:

Das durch A1 festgelegte Communication-Element wird mit Hilfe der dort eingetragenen Priorität an dem ihm zustehenden Platz in der Warteschlange, deren Nummer ebenfalls im CE enthalten ist, eingekettet und die zugehörige Betreuungstask wird alarmiert. Die Aktivierung der Betreungstask für diese Warteschlange ('I/O-Task') erfolgt mit deren Eigenpriorität.

Die Warteschlange kann nicht überlaufen. Die im CE gespeicherte Warteschlangennummer wird dahingehend geprüft, ob eine zugehörige Betreuungstask überhaupt vorhanden ist. Im Fehlerfalle wird die Meldung 'WRONG LDN' abgesetzt und im CE der Wert von RECLEN auf Null gesetzt.

This page intentionally left blank.

RTOS-UH Floppy/Winchester-Filesystem

Es wird mit logischen Blöcken gearbeitet, deren jeweilige physikalische Position auf dem Speichermedium durch die 'Untergliederungsdaten' errechenbar ist. Bis auf den Block No. 0 sind die Blöcke von jeweils gleichem Aufbau. Block 0 beginnt auf Track 00, Sektor 1 und Seite 0.

Die leere Diskette muß im Block 0 wie folgt vorformatiert sein:

$00	HDDRV	11 00	Kennungskopf des UH-Fman zur Unterscheidung
$02		00 00	z.B von Boot-Disketten, CP/M-Disketten etc.
$04	SODSID	00 0x	Single Or Double-SIDed x=0 single x=1 double
$06	FBLEN	xx xx	Anzahl phys. Bytes pro Block, immer n*256
$08	NOBLPT	xx xx	2-er Logarithmus aus Blockzahl/Track/Seite Also 0 -> 1 Bl/Track, 1 -> 2 Bl/Track etc
$0A	NOSEPB	xx xx	Anzahl Sektoren pro Block, Sektorinkrement
$0C	HDBLNO	xx xx	HeaDer BLock Nummer - 1
$0E	ABMLEN	xx xx	Anzahl der insgesamt vorhandenen Blöcke
$10	ABMIDX	aa aa	ABM-Start-Index zum Auffinden der ABM
$12	DIRSTR	dd dd	Directory-Start-Index zum Auffinden des Dir.
$14	DIRLEN	xx xx	Directory-Länge = max. mögl. Anzahl Files
$16	DIRLEE	xx xx	Länge des Einzeleintrages/File im Directory Mindestens Namenslänge+8 (normal:16)
$18	NAMLEN	xx xx	Namenslänge (max. Anz. Bytes) im Directory (normal:8)
$1A	RES1	00 00	Reserviert
$1C	RES2	00 00	Reserviert
$1E	RES3	00 00	Reserviert
$20	LABEL	zz zz	'Label'-Text der Diskette (typ. 80 Bytes)
.....			
$aaaa		.. ..	Allocation-byte-map 'ABM'. Muß initialisiert werden mit den ersten 2 Bytes $FF,$FF und allen weiteren Bytes jeweils Null. Es sind (4*Anzahl Blöcke) Bytes anzulegen.
$dddd		.. ..	Directory. Muß initialisiert werden mit: Erster Eintrag Name **FREE** (mit Blanks auf Namenslänge gefüllt) und alle weiteren (total: DIRLEN*DIRLEE) Bytes jeweils Null.
....			Eventueller Rest bleibt ungenutzt.

Die Datenblöcke benötigen nur die Hardware-Initialisierung. Die Struktur ist hier nur zur Information angegeben:

$00	FORBLK	xx xx	Folgeblock (0 falls letzter Block des Files)
$02	BACBLK	xx xx	Vorgängerblock (0 falls erster Block)
$04	MINIDX	aa aa	Index des ersten Nutdatenbytes
$06	ACTIDX	bb bb	Aktueller Lese-/Schreibindex
$08	WRTIDX	cc cc	Writer-Index. Schleppzeiger von ACTIDX
$0A	WMXIDX	dd dd	Indexgrenze (Letztes Byte des Blocks+1)
...			
$aaaa			Erstes Nutzdatum
...			
$bbbb			(Zufälliger) aktueller Zeiger
...			
$cccc			(Zufällige) Schreiber-Schlepposition
...			
$dddd-1			Letztes Nutzdatum

In Notfällen wäre also auch ein Lesen der Files ohne Directory möglich, wenn die Untergliederung (Blocklänge etc.) bekannt ist. Allerdings ist der Filename dann nicht feststellbar.

Das Betriebssystem übernimmt die Betreuung von Warteschlangen an E/A-Bausteinen mit begrenzter Übertragungsgeschwindigkeit, wie z.B. UART (ACIA) oder solchen digitalen Ein-/Ausgaben, bei denen zwischen den Transfers aus anderen Gründen Wartephasen anfallen (z.B. Floppy-Koppler o.ä.). Für rein elektronische und damit aus der Sicht des digitalen Prozesses beliebig schnell mögliche E/A ist das Betriebssystem nicht zuständig, da keine Wartephasen anfallen, die einen Taskwechsel sinnvoll werden lassen.

Eine Task, die eine Ein-/Ausgabe beginnt, muß sich zuvor in den Besitz eines 'Communication-Elements' (CE) bringen - wenn sie nicht noch ein 'leeres' von einer vorhergehenden E/A besitzt. Dafür ist die Operation 'FETCE' vorgesehen.

Das CE wird von der Task parametriert, z.B. durch Einsetzen der Warteschlangen-Nummer ('LDN'), des Übertragungsmodes, der Satzlänge, des File-Namens usw.

Das gefüllte CE wird von der Task mit Hilfe der Operation 'XIO' dem Betriebssystem angeboten. 'XIO' bringt die Task in den Wartezustand 'I/O?', sofern bei der Parametrierung des CE das Wartebit im Übertragungsmode gesetzt wurde. Wenn das Freigabebit ('Release'-bit) von der Task gesetzt wurde, ist der Aufruf von 'XIO' die letzte zulässige Operation der Task mit diesem CE, da es nach Abarbeitung in der Warteschlange automatisch als freier Speicher in die Verwaltung von RTOS-UH zurückkehrt.

Wenn das Freigabebit nicht gesetzt wurde, muß die Task in geeigneter Weise mit dem 'CE' weiter verfahren. Eine Neuparametrierung ist erst erlaubt, nachdem das CE abgearbeitet wurde. War beim Aufruf von 'XIO' das Wartebit gesetzt, so kann die Task nach dem 'XIO' sofort neu über das CE verfügen. Andernfalls ist mit der Systemfunktion 'IOWA' (I/O-Wait) das Ende der I/O-Operation abzuwarten. Ggf. kann noch vor einer Neuparametrierung nach dem 'IOWA' der Rückmeldecode im CE analysiert werden.

Ein CE wird auf drei mögliche Arten wieder zu freiem Speicherraum:

1. Aufruf von 'XIO' mit gesetztem Freigabebit.
2. Die besitzende Task wird terminiert oder beendet sich selbst.
3. Die Task ruft die Operation 'RELCE' für dieses CE auf.

Der Aufruf von 'RELCE' (Release CE) ist immer möglich, solange die Task Besitzer des CE ist, also auch während eines noch laufende E/A-Vorganges über dieses CE. 'RELCE' beeinflußt nicht die laufende oder durch 'XIO' gerade veranlaßte E/A-Operation.

Bei der Terminierung einer Task ist zwischen Ein- und Ausgabe zu unterscheiden:

- Bei der Ausgabe verbleiben mit 'XIO' bereits in die Ausgabewarteschlange gebrachte CEs dort, es wird nur das Freigabebit gesetzt und die Task-ID im CE gelöscht.

- Bei der Eingabe werden die CEs, die noch nicht in Bearbeitung durch die Betreuungstask der Warteschlange sind, aus der Schlange genommen und zu freiem Speicher umgewandelt. In Bearbeitung befindliche CEs verbleiben bei der Betreuungstask (die angefangene Eingabe muß also regulär zu Ende geführt werden) bis sie abgearbeitet sind und werden dann zu freiem Speicher umgewandelt.

Das CE ist ein wichtiger Bestandteil der Ein-/Ausgabe im Betriebssystem RTOS-UH. Dabei ist der genaue Aufbau nur für den Assemblerprogrammierer interessant. Für den Hochsprachanwender übernehmen Compiler und Laufzeitsystem die Parametrierung der CE's.
In der folgenden Tabelle sind in der linken Spalte die Displacements der einzelnen Parameter relativ zu A1, dem Zeiger auf das CE nach 'FETCE', und daneben kurz ihre Bedeutung angegeben:

$00 - $11	Verwaltungszeiger, nicht verändern!	
$12 - $15	TIDO	Task-ID of owner, "
$16 - $1D	Verwaltungszeiger, nicht verändern! (Queue-links)	
$1E - $1F	PRIO	Plazierungsprio. in der Queue, darf verändert werden. Defaultwert ist die aktuelle Laufprio. der Task.
$20 - $23	BUADR	Buffer-adress. Darf verändert werden.
$24 - $25	RECLEN	Blocklänge (Anzahl Bytes), I/O-Task gibt eine Null zurück, falls I/O nicht ausführbar.
$26	STATIO	Statusbyte
$27	LDNIO	LDN der Warteschlange, für die das CE bestimmt ist oder in der es steht.
$28 - $29	MODE	Betriebsart, Endebedingung etc.
$2A - $2B	DRV	Drive-no. oder ACIA-mode
$2C - ...	FNAME	File-name oder Pathlist, Autostop durch $FF

Bedeutung der einzelnen Bits im Mode-Wort (ON='1', OFF='0'):

1. Byte	$80	MODMWA	Suspend (Wait) until ready
	$40	MODMOU	Output-direction
	$20	MODMCR	Auto-stop after transmission of a Carriage-return
	$10	MODMLF	Auto-stop after transmission of a Line-feed
	$08	MODMEO	Auto-stop after transmission of a EOT
	$04	MODMSC	Suppress Command-inp. (ACIA), Ctrl A,B,C ohne Wirkung
	$02	MODMNE	No echo on input (ACIA, SCC)
	$01	MODBIN	binärer Transfer

Das 2. Byte im Mode-Wort hat nicht bei allen Stationen eine Bedeutung, am dichtesten belegt ist es für den Floppy-Filehandler.

2. Byte	$80	IOCEF	Rückmeldebit bei Eingabe: End-of-file
	$40	NERR	No Error messages
	$20	EXCLU	Exclusiv access
	$00 ..	$1F	Betriebsbefehle lt. Tabelle:
		$00	READ/WRITE 'OLD' File
		$01	ERASE the File
		$02	
		$03	
		$04	RETURN the File
		$05	
		$06	CLOSE the File
		$07	READ/WRITE 'ANY'
		$08	REWIND the File
		$09	APPEND the File
		$0A	$0B
		$0C	(list) FILES (CE in the CE !)
		$0D	(list) FREE (CE in the CE !)
		$0E	(list) DIRectory (")
		$0F	
		$10	SYNC (update Dir., physic.)
		$11	
		$12	
		$13	SEEK (change position in file)
		$14	SAVE next pos. to write to a file (SEEK's counterpart)
		$15	(special) REWIND ('ANY')
		$16	
		$17	FORMAT single density
		$18	FORMAT double density
		$19	MaK DIRectory
		$1A	ReMove DIRectory

Bedeutung der Bits im Statusbyte 'STATIO':

$01	--	Reserviert
$02	STABRE	RTOS-UH soll oder will CE auflösen, sobald E/A fertig ist. Im Laborjargon als 'Verschrottungsbit' bekannt. Wird vom Nutzer z.B. vor dem 'XIO' gesetzt (dann ist das CE mit dem 'XIO' für ihn gestorben) oder implizit durch 'RELCE', 'TERMI', 'TERME'.
$04	--	Reserviert.
...	--	"
$08	STABFL	Flag zur freien Verwendung des Nutzers

This page intentionally left blank.

Der UH-Assembler ist auf das UH-PEARL-System abgestimmt, es können mit Hilfe des Laders problemlos auch assemblercodierte Unterprogramme in die PEARL-Umgebung integriert werden. Die Übergabe der Programmkontrolle an das Unterprogramm (UP) erfolgt mit Hilfe virtueller Befehle, die vom PEARL-Laufzeitsystem (Hyperprozessor) exekutiert werden. Wir betrachten dazu den Fall

```
CALL ABCD(para-list);
```

wobei ABCD ein Assemblerunterprogramm sein soll; Das Symbol ABCD muß im aufrufenden PEARL-Programm als ENTRY GLOBAL spezifiziert sein. Auf der Assemblerseite wird ABCD durch Voranstellung des Zeichens '>' global deklariert. Das PEARL-Run-time betritt das UP im virtuellen Mode an der Stelle >ABCD, unser UP muß also in jedem Fall mit einem virtuellen Befehl beginnen, wobei dies ggf. der Befehl V0 zum Abschalten des Hyperprozessors sein kann. Die weiteren Randbedingungen beim Eintritt sehen wie folgt aus:

A0 zeigt auf die Stelle an der das Paramterlink beginnt.
D2 enthält das Register A5 (='WL') des Aufrufers.
A5 ist noch identisch zu D2.

**** Die Register A4,A5,A6,A7 dürfen im Laufe des UP nicht vom Programmierer verändert werden. Kurzzeitige Veränderung ist zwar in gewissen Fällen möglich, wird aber nicht empfohlen. A7 ist mit max. 2 BSR-Level (8 Bytes) belastbar, es muß beim UP-Austritt oder vor der Benutzung irgendwelcher virtuellen Befehle unbedingt wieder auf dem Eingangswert stehen.
Bitte denken Sie auch daran, daß es Ihre eigene Aufgabe ist, dafür zu sorgen, daß das UP REENTRANT ist. Es dürfen also keine statisch allokierten Objekte im UP verändert werden, wenn Sie diese Bedingung einhalten wollen.
Im folgenden sollen 2 Fälle unterschiedlicher Schwierigkeit betrachtet werden:

Fall A: Es sollen weder Parameter noch Funktionswerte übergeben werden. Die Prozedur benötigt außer einigen wenigen Registern keinen eigenen Arbeitsspeicher. Ein Transfer von Werten ist über globale PEARL-Objekte dennoch möglich.

Auf der PEARL-Seite generiert der Compiler aus CALL ABCD; die Sequenz:

```
PROC  >ABCD          (PROC=V18)
EPAR                 (EPAR=V19)
```

Auf der Assembler-UP-Seite programmieren wir:

```
Evtl. 'module-head' für RTOS (s.D-III)
>ABCD   V0                   Hyperproc. off
        CMPI.B  =19,(A0)     Vergl.auf EPAR
        BNE     ...          Error - Zweig
        ....                 68000-Usr-code
```

```
*       D0..D7 zur freien Verfügung
*       A1..A3        "
        ....
        JMP     2(A0)         Rücksprung HP
        ....
```

Das ist die schnellste Form des Anschlusses überhaupt. Notfalls kann auf das Überprüfen der leeren Parameterliste (CMPI.B ...) auch noch verzichtet werden. Wenn Sie z.B. in dem UP Floatingbefehle benötigen, so muß zuvor der Hyperprozessor wieder angeworfen werden. Dies gelingt mit Hilfe des dem UH-Assembler bekannten Befehles TV, der nur ein bequemer Mnemo für den Trap 14 ist.

Beim Zugriff auf in der PEARL-Welt global deklarierte Felder sind einige Besonderheiten zu beachten, da dem eigentlichen Variablenfeld im Speicher ein Vorspann (Feldbeschreibungsblock) vorangestellt ist. Aus

DCL A(bound1,bound2,bound3) ... GLOBAL wird im Speicher

```
>A      DC.W  bound3                          2 Bytes
        DC.W  bound2                          2 Bytes
        DC.W  bound1                          2 Bytes
        DC.L  offset+adr                      4 Bytes
        DS    (Speicherbereich des Feldes)  size Bytes
```

Bei weniger als 3 Dimensionen entfällt die Ablage der nicht angegebenen Feldgrenzen. Die Werte von bound1 etc. sind mit der 'Global+offset'-option des UH-Assemblers ohne Probleme adressierbar. Die 32-bit-zahl 'offset+adr' gibt die physikalische Adresse an, auf der das Feldobjekt A(0,0,0) (welches nicht existiert) stehen würde. Damit wird die Indexformel beim Feldzugriff durch den Compiler (und den Assemblerprogrammierer) schneller und kürzer. Bei Unklarheiten ist ein Studium des vom Compiler generierten Codes (/*+P*/-option) sehr zu empfehlen.

Fall B: Es sollen Parameter und Funktionswerte übergeben werden. Außerdem nehmen wir an, daß unser UP zusätzlichen internen Arbeitsspeicher benötigt, dabei aber 'reentrant' sein soll. Wir betrachten wiederum zunächst die PEARL-Seite:

```
SPC ABCD ENTRY(FIXED,FLOAT IDENT) RETURNS(FLOAT(55))
                                                GLOBAL;

......
```

```
Z=ABCD(3,Y);          /* Der Compiler generiert hieraus: */

PROC  >ABCD                 (V18=PROC)
INVW  Xcon=3                (V14=INVW)
VARF  Y                     (V12=VARF)
VARD  resultcell            (V13=VARD)
EPAR                        (V19=EPAR)
...
MOVD  resultcell,Z         (MOVD ist ein Macro, 2x MOVE.L)
```

Beim Studium des Codes wird erkennbar, daß Funktions-UPs sich nur durch einen zusätzlich angehängten Übergabeparameter von den durch 'CALL' aufrufbaren UPs unterscheiden. Der Rückgabewert muß in eine vom Compiler beschaffte Zelle geschrieben werden.

Vor der Codierung unseres Assembler-UP muß zunächst bestimmt werden, wieviel Prozedurworkspace 'PWSP' gebraucht wird. In diesem PWSP müssen auch die Parameterwerte oder - bei IDENT - die vier Byte langen Adressen der Objekte untergebracht werden. Sollen Felder übergeben werden, so müssen auch Kopien der Feldbeschreibungsblöcke (s.o.) darin Platz finden.

In unserem Beispiel sei angenommen, daß das UP intern das 4 Byte lange Objekt HILF1 und das 8 Byte lange Objekt HILF2 benötigt. Wir codieren:

```
-- ggf. module-head nach Beschreibung D-III --
*       Vereinbarungen zur besseren Lesbarkeit des  Progr:
RETN    OPD     $4E4C       Opcode definition   =  TRAP 12
ENTR    OPD.V   29          Opcode proc-entry  (Hyperproc.)
INVW    OPD.V   14          Invariant 16 bit  fixed  =  V14
VARF    OPD.V   12          Variable Float 32 bit    =  V12
VARD    OPD.V   13          Variable Float 64 bit    =  V13
EPAR    OPD.V   19          End of parameter Xfer    =  V19
*       PWSP-Allocation
PAR1    EQU     0              2 bytes positioned to  0(A5)
PAR2    EQU     PAR1+2         4 bytes positioned to  2(A5)
WERT    EQU     PAR2+4         4 bytes (adr) pos. to  6(A5)
HILF1   EQU     WERT+4         4 bytes (internal) to 10(A5)
HILF2   EQU     HILF1+4        8 bytes (internal) to 14(A5)
WSPSZ   EQU     HILF2+8        Total Size of  Procedure-WSP
*
>ABCD   ENTR    WSPSZ.L      Fetch  storage  from 'RTOS-UH'
        INVW    PAR1.X       xfer 16-bit FIXED  by  'value'
        VARF    PAR2.Z       xfer 32-bit FLOAT  by  'Ident'
        VARD    WERT.Z       xfer 64-bit FLOAT  by  'Ident'
        EPAR                 End of param.,  hyperproc  off
        ....
*       reeller 68000-code.  D0...A3 sind  frei verfügbar.
*       A5  zeigt auf Ablageplatz  des  Wertes  von  PAR1.
```

```
*       Der Zugriff auf die Objekte:

        MOVE.L  =$01400000,HILF1(A5)  oder (gleichwertig):
        MOVE.L  =$01400000,HILF1.X
        MOVE    PAR1.X,D4   Die Zahl 3 des obigen Bsp.->D4
        ....
        MOVEA.L PAR2.X,A0   Zeiger laden, da  IDENT  xfer!
        MOVE.L  =$01400000,(A0)  Wert-> 'Y' des Beispieles
        ....
        MOVEA.L WERT.X,A0   Zeiger auf Ergebniszelle laden
        MOVE.L  HILF2.X,(A0)+  xfer Bytes  1..4   Ergebnis
        MOVE.L  HILF2+4.X,(A0)  ''         5..6      ''
        ....
*       Bsp. letzte Sequenz unter Verwendung des Hyperproc
        TV                      Hyperproc hier  'anwerfen'
        V1      WERT.Z,HILF2.X  (Reihenfolge: dest,source)
        ....                    evtl. andere virt. Befehle
        V0                      Hyperproc  hier abschalten
        ....                    evtl.  weiterer  Realcode
*       Bsp. zum Verlassen des UP, nur im  Realcode moegl.
        ....
        RETN                    Realer Befehl, exit des UP
        END
```

Die Verwendung des V1-Befehles, der auch als MOVU OPD.V 1 hätte deklariert werden können, dient nur der Demonstration, da der Befehl inzwischen von den Compilern nicht mehr benutzt wird. (s. Liste) .MERKE: Hilfszellen und per 'value' übergebene Objekte werden mit .X oder (nur bei realen Befehlen möglich) ..(A5) adressiert. Durch Zeiger vertretene Objekte werden bei realen Befehlen durch Laden eines Adressregisters und bei virtuellen Befehlen durch die .Z -adressierung erreicht. Bei den Parameterübergabebefehlen im Kopf des UP wird durch .Z der Ident- und durch .X der Valuemode der Übergabe festgelegt. Bei Identmode darf das Objekt der Aufrufseite nicht invariant sein, sonst wird wie bei falschem Parameterdatentyp ein Laufzeitfehler angezeigt. Grundsätzlich kann man die Übergabe auch ohne den Hyperprozessor schaffen, wenn der Aufbau der Aufrufseite durch das UP selbst interpretiert wird.

Liste der Parameterübergabebefehle und Hinweise zur Verwendung.
.....

A. Datentypen FIXED,FLOAT skalar.

INVD	V17	Invariant	Float (55)	= 8	Byte Float
INVF	V16	Invariant	Float (23)	= 4	Byte Float
INVW	V14	Invariant	Fixed (15)	= 2	Byte Fixed
INVX	V15	Invariant	Fixed (31)	= 4	Byte Fixed
VARD	V13	Variable	Float (55)	= 8	Byte Float
VARF	V12	Variable	Float (23)	= 4	Byte Float
VARW	V10	Variable	Fixed (15)	= 2	Byte Fixed
VARX	V11	Variable	Fixed (31)	= 4	Byte Fixed

B. Datentypen CHAR, BIT(2 typen!), DUR, CLOCK, STRUCT skalar.

MPXF OPD.V 139 Miscellaneous parameterxfer 2 Op.

Anwendung: MPXF dtyp,object.X oder ...object.Z

Die Zelle DTYP (2/4 Byte) muß die Datentypbeschreibung enthalten.

1. Byte in dtyp:

nur bei CHAR und BIT (sonst 0): Objektlänge.

2. Byte in dtyp:

```
$08  = CHAR        ;  $09  = INV CHAR
$0A  = BIT(1..16);  $0B  = INV BIT(1..16)
$0C  = BIT(17..32); $0D  = INV BIT(17..32)
$0E  = DURATION    ; $0F  = INV DURATION
$10  = CLOCK       ; $11  = INV CLOCK
$12  = STRUCT      ; $13  = INV STRUCT
```

3./4. Byte in dtyp:

Nur bei STRUCT nötig: Number of Bytes of obj.

Beispiel:

```
MPXF  OPD.V 139
      ....
      MPXF  C7,TEXT.X     'Value'-übergabe  String.
      MPXF  B16,MASK.Z    'Ident'-übergabe  Bitmask
      ....
C7    DC.B  7,$09         CHAR(7)  invariant object.
B16   DC.B  16,$0A        BIT(16)  variable  object.
      ....
*     Für  MASK müssen 4 Byte (Adresse!)   und  für
*     TEXT  7  Bytes im   PWSP   vorgesehen   sein.
*     Bei 'value'-übergabe findet ggf.  eine Anpas-
*     sung der CHAR-Längen statt.
```

C. Übergabe von Feldern.

C1. Arrays der Typen FIXED,FLOAT,DURATION,CLOCK,STRUCT.

```
ARNS  OPD.V  122       'Array no string' 3 Operanden.
      ....
      ARNS  offs+adr-cell,boundlistcells,descr.mask
      ....
      ARNS  OFFS.X,BOUNDS.X,$0812.X     (2dim DUR-Feld)
*
```

```
*       Für OFFS  sind 4  Bytes bereitzustellen, für die
*       Liste der bounds (2*Anz.Dimensionen)  Bytes. Die
*       Feldgrenzen werden in der Reihenfolge des  Feld-
*       beschreibungsblockes abgelegt.  Auf  OFFS  steht
*       die (fiktive) Speicheradresse des  Feldelementes
*       (0) bzw. (0,0) oder (0,0,0). Wenn OFFS unmittel-
*       bar hinter BOUNDS liegt, entsteht somit ein kom-
*       pletter neuer Feldbeschreibungsblock.
```

Die 'descriptionmask' enthält Informationen über die Anzahl der Dimensionen und den Datentyp wie folgt:

Typ	1dim, (inv)	2dim, (inv)	3dim, (inv)
FIXED(15)	$0401,$0541	$0801,$0941	$0C01,$0D41
FIXED(31)	$0402,$0542	$0802,$0942	$0C02,$0D42
FLOAT(23)	$0404,$0544	$0804,$0944	$0C04,$0D44
FLOAT(55)	$0408,$0548	$0808,$0948	$0C08,$0D48
DURATION	$0412,$0552	$0812,$0952	$0C12,$0D52
CLOCK	$0416,$0556	$0816,$0956	$0C16,$0D56
STRUCT	$0415,$0555	$0815,$0955	$0C15,$0D55

Bekanntlich ist im UH-PEARL der Feldtransfer nur im 'ident'mode möglich, so daß es keine weitere Fallunterscheidung wie bei den Skalaren gibt.

Bei den Verbundobjekten (STRUCT) folgt dem ARNS noch eine Längenüberprüfung LTST (=V72) als weiterer Parameterübergabebefehl. Er hat einen Operanden, der unmittelbar die Anzahl Bytes, aus der die Struktur besteht, representiert.

```
ARNS   ...,...,$0415.X  Übergabe Verbundfeld.
LTST   25.X             Verbundtyp besteht aus 25 Bytes.
```

C2. Arrays der Datentypen BIT und CHAR.

```
ARBS   OPD.V  123       'array bitstring',  3 Operanden
ARCS   OPD.V  124       'array char.string',3 Operanden

       ARCS   offs+adr-cell,boundlist-cells,descr.mask
       ARBS                      ''

       ARCS   OFFS.X,BOUNDS.X,$0812.X = ..(,) CHAR(18)
       ARBS   OFF2.X,BOU2.X,$0C09.X   = ..(,,)  BIT(9)
```

Es gelten sinngemäß die gleichen Bedingungen wie bei der Instruktion 'ARNS'; im Gegensatz zum 'ARNS' enthält die 'description-mask' nun im rechten Byte die Länge in bytes (char) oder in bits (bit) des Feldelementes. Im Einzelfall ist wie folgt zu kodieren:

Typ	Bef.	1dim, (inv)	2dim, (inv)	3dim, (inv)
BIT(xx)	ARBS	$04xx,$05xx	$08xx,$09xx	$0Cxx,$0Dxx
CHAR(yy)	ARCS	$04yy,$05yy	$08yy,$09yy	$0Cyy,$0Dyy

Die virtuellen Maschinenbefehle des Laufzeitsystemes bilden den sogenannten 'Hyperprozessor'. Die Benutzung der Befehle ist zwar auch dem Assemblerprogrammierer möglich, doch wird keine Gewähr für langfristige Unveränderlichkeit des Hyperprozessors gegeben. Im Zuge von Compilerverbesserungen können immer auch Veränderungen des Befehlssatzes auftreten. Man überzeuge sich daher genau, ob die aktuelle Variante noch alle benutzten Befehle enthält.

No.	XW	Mnemo	Bemerkungen	
V0	-	TOREAL	Abschalten des Hyperprozessors	*
V1	2	MOVU	Nicht für Neuentwicklung (Move 8 bytes)	*
V2	1	(ADDF)	Nicht für Neuentwicklung (Add Float 23)	*
V3	1	(ADDD)	Nicht für Neuentwicklung (Add Float 55)	*
V4	1	(SUBF)	Nicht für Neuentwicklung (Sub Float 23)	*
V5	1	(SUBD)	Nicht für Neuentwicklung (Sub Float 55)	*
V6	1	(MULX)	Nicht für Neuentwicklung (Mul Fixed 31)	*
V7	1	(MULD)	Nicht für Neuentwicklung (Mul Float 55)	*
V8	1	(DIVX)	Nicht für Neuentwicklung (Div Fixed 31)	*
V9	1	(DIVD)	Nicht für Neuentwicklung (Div Float 55)	*
V10	1	VARW	Param. XFER: Variable Fixed 15	*
V11	1	VARX	Param. XFER: Variable Fixed 31	*
V12	1	VARF	Param. XFER: Variable Float 23	*
V13	1	VARD	Param. XFER: Variable Float 55	*
V14	1	INVW	Param. Xfer: Konstante Fixed 15	*
V15	1	INVX	Param. XFER: Konstante Fixed 31	*
V16	1	INVF	Param. XFER: Konstante Float 23	*
V17	1	INVD	Param. XFER: Konstante Float 55	*
V18	1	CALL	Call procedure. (Im Comp. als PROC)	*
V19	-	EPAR	End of Parameterlist	*
V20	2	SHFS	Shift 1 TO 16 BIT Object	*
V21	2	SHFL	Shift 17 to 32 BIT Object	*
V22	-	CVXF	Nicht für Neuentwicklung, Fix31 to Flo23	*
V23	-	CVWD	Nicht für Neuentwicklung, Fix15 to Flo55	*
V24	-	CVXD	Nicht für Neuentwicklung, Fix31 to Flo55	*
V25	-	CVFD	Nicht für Neuentwicklung, Flo23 to Flo55	*
V26	4	CATC	Concatenation von 2 CHAR-strings	*
V27	3	PUT	Eröffnung einer Liste für PUT	*
V28	3	GET	Eröffnung einer Liste für GET	*
V29	1	ENTR	Prozedureintritt mit PWSP-Erzeugung	*
V30	2	EFR2	E-Format mit 2 Parametern	*
V31	3	EFR3	E-Format mit 3 Parametern	*
V32	1	PAGE	PAGE-Format	*
V33	1	XFOR	X-Format	*
V34	1	RLFO	Remote-Left-Bracket in Format	*
V35	1	EALW	E/A of Fixed15	*
V36	1	EALX	E/A of Fixed31	*
V37	1	EALF	E/A of Float23	*
V38	1	EALD	E/A of Float55	*
V39	1	EADU	E/A of Duration	*
V40	1	EACL	E/A of Clock	*
V41	-	LBRK	Left Bracket in Format	*
V42	-	RBRK	Right Bracket in Format	*
V43	-	FESP	Format-End-specification	*
V44	1	FACT	Format Wiederholfaktor	*

V45	1	AFOL	A-Format	*
V46	-	LIFO	List-Format	*
V47	1	SKFO	SKIP-Format	*
V48	3	OPN3 **	OPEN 3 Param. (BY IDF(..))	*
V49	1	RWND **	REWIND Dation and open	*
V50	1	SYNC **	Synchronize Dation	*
V51	1	APND **	Append to a File	*
V52	1	EORL	EOR with 32 Bit Obj.	*
V53	2	CSHL	Cyclic Shift 17 to 32 Bit	*
V54	2	IOBS	I/O Bit-string	*
V55	2	CSHS	Cyclic Shift 1 to 16 Bit	*
V56	2	SEEK **	Seek a position in a file	*
V57	2	SAVP **	Save a position in a file	*
V58	-	----	Nicht mehr besetzt	*
V59	3	NEBS	Not equal for Bit-strings	*
V60	4	NECS	Not equal Character-string	*
V61	-	----	Nicht mehr besetzt	*
V62	-	----	Nicht mehr besetzt	*
V63	1	EQUD	Equal long Float	*
V64	1	SUSE	Suspend external (given Task)	*
V65	-	----	Nicht mehr besetzt	*
V66	1	NEQD	Not equal long Float	*
V67	-	----	Nicht besetzt	*
V68	-	----	Nicht besetzt	*
V69	-	----	Nicht besetzt	*
V70	-	----	Nicht besetzt	*
V71	-	----	Nicht besetzt	*
V72	1	LTST	Length-test (Struct-param xfer)	*
V73	1	(LTHF)	Nicht für Neuentwicklung (LT Float(23))	*
V74	1	(LTHD)	Nicht für Neuentwicklung (LT Float(55))	*
V75	-	----	Nicht mehr besetzt	*
V76	-	----	Nicht mehr besetzt	*
V77	-	----	Nicht mehr besetzt	*
V78	-	----	Nicht mehr besetzt	*
V79	-	----	Nicht mehr besetzt	*
V80	-	----	Nicht mehr besetzt	*
V81	-	----	Nicht mehr besetzt	*
V82	-	----	Nicht mehr besetzt	*
V83	1	POWW	Power Fixed(15)	*
V84	1	POWX	Power Fixed(31)	*
V85	1	POWF	Power Float(23)	*
V86	1	POWD	Power Float(55)	*
V87	1	TERM	Terminate (given Task)	*
V88	1	PREV	Prevent (given Task)	*
V89	1	CONT	Continue (given Task)	*
V90	1	LIMV	Line-marker in virtual environment	*
V91	1	ODAT	Open Dation (no operation)	*
V92	1	CDAT	Close Dation (if closable)	*
V93	1	WCON	(When)... Continue	*
V94	1	TCON	(Timed) ..Continue	*
V95	2	ACTI	Activate with priority	*
V96	2	WACT	(When)..activate with priority	*
V97	2	TACT	(Timed)..activate with priority	*
V98	-	WRES	(When) .. Resume (own Task)	*
V99	-	TRES	(Timed) .. Resume (own Task)	*
V100	-	STSC	Start schedule-definition	*

V101	1	ATCL	AT (clock) schedule param. set	*
V102	1	AFTR	AFTER (duration) schedule param. set	*
V103	1	ALLD	ALL (duration) schedule param. set	*
V104	1	UNIL	UNTIL (clock) schedule param. set	*
V105	1	DUDU	DURING (duration) schedule param. set	*
V106	1	WHEV	WHEN (event) schedule param. set	*
V107	3	WRIT	WRITE-Instruction	*
V108	3	READ	READ-Instruction	*
V109	-	----	Nicht mehr besetzt	*
V110	-	----	Nicht mehr besetzt	*
V111	1	EOLI	End I/O-List (escape-label)	*
V112	-	RFEN	Remote-format end	*
V113	2	EACS	E/A Character-string	*
V114	3	FFOR	F-Format	*
V115	-	----	Nicht besetzt	*
V116	-	----	Nicht besetzt	*
V116	-	----	Nicht besetzt	*
V118	4	EQBS	Equal Bit-strings	*
V119	4	EQCS	Equal char. strings	*
V120	3	MVBS	Move Bit-string	*
V121	3	MVCS	Move character-string	*
V122	3	ARNS	Array-param.xfer 'no string'	*
V123	3	ARBS	Array-param.xfer 'bitstring'	*
V124	3	ARCS	Array-param.xfer 'char.string'	*
V125	2	BFOR	B-format	*
V126	2	DFOR	D-format	*
V127	2	TFOR	T-format	*
V128	-	----	Nicht mehr besetzt	*
V129	-	----	Nicht mehr besetzt	*
V130	-	----	Nicht mehr besetzt	*
V131	1	(MULF)	Nicht für Neuentwicklung (MUL Float(23))	*
V132	1	(DIVF)	Nicht für Neuentwicklung (DIV Float(23))	*
V133	1	EORW	Excl.Or 32 Bit (Fehlt in 68000 Hardw.)	*
V134	-	(CVWF)	Nicht für Neuentwicklung (Fix15 to F123)	*
V135	1	(LDAD)	Nicht für Neuentwicklung (Load acc.8 by)	*
V136	1	(STAD)	Nicht für Neuentwicklung (St. accu 8 by)	*
V137	1	DMYD	(Dummy)-Dation parameter xfer	*
V138	1	DMYI	(Dummy)-Interrupt parameter xfer	*
V139	2	MPXF	Miscellaneous Parameter X-fer	*
V140	-	(ABSF)	Nicht für Neuentwicklung (ABS accu F123)	*
V141	-	(ABSD)	Nicht für Neuentwicklung (ABS accu F155)	*
V142	-	(ENTI)	Nicht für Neuentwicklung (ENTIER)	*
V143	-	(ROUN)	Nicht für Neuentwicklung (ROUND)	*
V144	-	(SIGN)	Nicht für Neuentwicklung (SIGN)	*

Im Folgenden sind alle Befehle, die unsere Compiler erzeugen, in alphabetischer Reihenfolge aufgelistet.

Mnemo	XW	Operation	Bemerkung
ABAL	-	ADD.L A5,D0	Add base adr long
ABSD	-	JSR -72(A6)	
(ABSD)	-	V141	Nicht für Neuentwicklung (ABS accu Fl55)
ABSF	-	JSR -68(A6)	ABS Flo23
(ABSF)	-	V140	Nicht für Neuentwicklung (ABS accu Fl23)
AC-1	-	SUBQ =1,D0	
ACTI	2	V95	Activate with priority
ADDA	-	ADD.L A1,D0	
(ADDD)	1	V3	Nicht für Neuentwicklung (Add Float 55)
ADDD	1	Macro	LEA obj,A1 + JSR -20(A6)
(ADDF)	1	V2	Nicht für Neuentwicklung (Add Float 23)
ADDF	1	Macro	LEA obj,A1 + JSR -4(A6)
ADDI	2	ADDI.W =con,xxx	
ADDM	1	ADD.W D0,xxx	
ADDW	1	ADD.W	Native Add
ADDX	1	ADD.L	Native Add long
ADIL	1	ADD.L =xx,D0	Add immediate long
AD+2	-	ADDQ.L =2,A1	Quick adr. shifter
AFOL	1	V45	A-Format
AFTR	1	V102	AFTER (duration) schedule param. set
ALLD	1	V103	ALL (duration) schedule param. set
ANDW	1	AND.W	Native AND Word
ANDL	1	AND.L	Native AND Long
ANIW	1	ANDI.W =xx,D0	
ANIL	1	ANDI.L =xx,D0	
APND	1	V51 **	Append to a File
ARBS	3	V123	Array-param.xfer 'bitstring'
ARCS	3	V124	Array-param.xfer 'char.string'
ARNS	3	V122	Array-param.xfer 'no string'
ATCL	1	V101	AT (clock) schedule param. set
BFOR	2	V125	B-format
BGE4	-	BGE.S $+4	
BGEL	1	BGE.L xxx	Conditioned branch ge
BGTL	1	BGT.L xxx	Conditioned branch gt
BLTL	1	BLT.L xxx	Conditioned branch lt
CALL	1	V18	Call procedure. (Im Comp. als PROC)
CASE	-	JMP -96(A6)	For CASE-construct
CATC	4	V26	Concatenation von 2 CHAR-strings
CDAT	1	V92	Close Dation (if closable)
CLRL	1	CLR.L xx	
CMPW	1	CMP.W	Native Compare to D0
CMPL	1	CMP.L	Native Compare long to D0
CONT	1	V89	Continue (given Task)
CSHL	2	V53	Cyclic Shift 17 to 32 Bit
CSHS	2	V55	Cyclic Shift 1 to 16 Bit
CVBL	-	Macro	Convert Bitstring to long SWAP D0 + CLR
CVFD	-	CLR.L D1	Convert Flo23 to Flo55
(CVFD)	-	V25	Nicht für Neuentwicklung, Flo23 to Flo55
CVWD	-	JSR -56(A6)	Convert Fix15 to Flo55
(CVWD)	-	V23	Nicht für Neuentwicklung, Fix15 to Flo55
CVWF	-	JSR -64(A6)	Convert Fix15 to Flo23

```
(CVWF)  -   V134      Nicht für Neuentwicklung (Fix15 to Fl23)*
CVWX    -   EXT.L D0  Convert Fix15 to Fix31                  *
CVXD    -   JSR   -60(A6)    Convert Fix31 to Flo55           *
(CVXD)  -   V24       Nicht für Neuentwicklung, Fix31 to Flo55*
CVXF    -   JSR   -52(A6)    Convert Fix31 to Flo23           *
(CVXF)  -   V22       Nicht für Neuentwicklung, Fix31 to Flo23*
D0A0    -   MOVEA.L  D0,A0   Quickload of adr-reg (optim.)    *
D0A1    -   MOVEA.L  D0,A1                ''                  *
D0*2    -   ADD D0,D0   Verdoppel D0                          *
D0*4    -   LSL =2,D0   Vervierfache D0                       *
D0*8    -   LSL =3,D0   Verachtfache D0                       *
DFOR    2   V126      D-format                                *
DISA    -   TRAP $A034  Disable Interrupt                     *
DIVD    1   Macro     LEA obj,A1  +  JSR    -32(A6)           *
(DIVD)  1   V9        Nicht für Neuentwicklung (Div Float 55) *
DIVF    1   Macro     LEA obj,A1  +  JSR    -16(A6)           *
(DIVF)  1   V132      Nicht für Neuentwicklung (DIV Float(23))*
DIVW    1   DIVS      Native divide                           *
DIVX    1   Macro     LEA obj,A1  +  JSR    -48(A6)           *
(DIVX)  1   V8        Nicht für Neuentwicklung (Div Fixed 31) *
DMYD    1   V137      (Dummy_-Dation parameter xfer           *
DMYI    1   V138      (Dummy)-Interrupt parameter xfer        *
DSCO    -   SUBQ =1,D2   Decrement Shift count one            *
DUDU    1   V105      DURING (duration) schedule param. set   *
EACL    1   V40       E/A of Clock                            *
EACS    2   V113      E/A Character-string                    *
EADU    1   V39       E/A of Duration                         *
EALD    1   V38       E/A of Float55                          *
EALF    1   V37       E/A of Float23                          *
EALW    1   V35       E/A of Fixed15                          *
EALX    1   V36       E/A of Fixed31                          *
EFR2    2   V30       E-Format mit 2 Parametern               *
EFR3    3   V31       E-Format mit 3 Parametern               *
ENAB    -   TRAP $A032    Enable Interrupt                    *
ENTI    -   JSR -76(A6)    ENTIER-Funktion                    *
ENTI)   -   V142      Nicht für Neuentwicklung (ENTIER)       *
ENTR    1   V29       Prozedureintritt mit PWSP-Erzeugung     *
EOLI    1   V111      End I/O-List (escape-label)             *
EORL    1   V52       EOR with 32 Bit Obj.                    *
EORW    1   V133      Excl.Or 32 Bit   (Fehlt in 68000 Hardw.)*
EPAR    -   V19       End of Parameterlist                    *
EQBS    4   V118      Equal Bit-strings                       *
EQCS    4   V119      Equal char. strings                     *
EQUD    1   V63       Equal long Float                        *
EQUF    1   Macro     Konstruktion mit CMP.L + versch. ff     *
EQUW    1   Macro     Konstruktion mit CMP.W + versch. ff     *
EQUX    1   Macro     Konstruktion mit CMP.L + versch. ff     *
EXIT    -   TRAP =1   Terminate own Task                      *
EX20    -   EXG  D2,D0    (in FOR .. BY variable .. REPEAT)   *
FACT    1   V44       Format Wiederholfaktor                  *
FESP    -   V43       Format-End-specification                *
FFOR    3   V114      F-Format                                *
FLFR    -   JSR -88(A6)     Float to fraction conv.  ('SEND') *
FRFL    -   JSR -92(A6)     Fraction to Floating conv.(TAKE)  *
GET     3   V28       Eröffnung einer Liste für GET           *
```

```
INVD     1     V17          Param. XFER: Konstante Float 55          *
INVF     1     V16          Param. XFER: Konstante Float 23          *
INVW     1     V14          Param. Xfer: Konstante Fixed 15          *
INVX     1     V15          Param. XFER: Konstante Fixed 31          *
IOBS     2     V54          I/O Bit-string                           *
ITS1     -     TRAP $A040   Index-test 1-dim.                        *
ITS2     -     TRAP $A042   Index-test 2-dim                         *
ITS3     -     TRAP $A044   Index-test 3-dim                         *
JRSI     -     JSR (A1)     Native pointered jump subroutine         *
JSR      1     JSR xxx      Native jump subroutine                   *
LBRK     -     V41          Left Bracket in Format                   *
LDAD     1     V135         Nicht für Neuentwicklung (Load acc.8 by)*
LDIL     1     MOVE.L =xx,D0  (Load immediate long)                  *
LEFA     1     LEA xx,A1                                             *
LIFO     -     V46          List-Format                              *
LIMR     -     TRAP $A036   Line-marker in real environment          *
LIMV     1     V90          Line-marker in virtual environment       *
LTHD     1     Macro        LEA obj,A1  +  JSR  -40(A6)              *
(LTHD)   1     V74          Nicht für Neuentwicklung ( LT Float(55))*
LTHF     1     Macro        LEA obj,A1  +  JSR  -36(A6)              *
(LTHF)   1     V73          Nicht für Neuentwicklung ( LT Float(23))*
LTST     1     V72          Length-test (Struct-param xfer)          *
MMBY     2     Macro        Move multibyte, schnelle DBF-Konstrukt. *
MOVB     ?     MOVE.B       Native Move byte                         *
MOVF     ?     MOVE.L       Native Move long                         *
MOVD     ?     Macro        Verschiedene Sequenzen native Code       *
MOVU     2     V1           Nicht für Neuentwicklung (Move 8 bytes)  *
MOVW     ?     MOVE.W       Native Move                              *
MOVX     ?     MOVE.L       Native Move long                         *
MPXF     2     V139         Miscellaneous Parameter X-fer            *
MULD     1     Macro        LEA obj,A1  +  JSR  -28(A6)              *
(MULD)   1     V7           Nicht für Neuentwicklung (Mul Float 55)  *
MULF     1     Macro        LEA obj,A1  +  JSR  -12(A6)              *
(MULF)   1     V131         Nicht für Neuentwicklung (MUL Float(23))*
MULU     1     MULU         Native Multiply 16 bit unsigned          *
MULW     1     MULS         Native Multiply 16 bit                   *
MULX     1     Macro        LEA obj,A1  +  JSR  -44(A6)              *
(MULX)   1     V6           Nicht für Neuentwicklung (Mul Fixed 31)  *
MVBS     3     V120         Move Bit-string                          *
MVCS     3     V121         Move character-string                    *
NEBS     3     V59          Not equal for Bit-strings                *
NECS     4     V60          Not equal Character-string               *
NEGW     1     NEG.W xx                                              *
NEGL     1     NEG.L xx                                              *
NEQF     1     Macro        Konstruktion mit CMP.L + versch. ff      *
NEQW     1     Macro        Konstruktion mit CMP.W + versch. ff      *
NEQX     1     Macro        Konstruktion mit CMP.W + versch. ff      *
NOT      1     NOT.W        Native NOT Word                          *
NOTW     1     NOT.W        Native NOT Word                          *
NOTL     1     NOT.L        Native NOT Long                          *
ODAT     1     V91          Open Dation (no operation)               *
OPN3     3     V48   **     OPEN 3 Param. (BY IDF(..))               *
ORW      1     OR.W         Native OR                                *
ORL      1     OR.L         Native OR long                           *
PAGE     1     V32          PAGE-Format                              *
```

```
PNTR    1     DC.W ...  Only Extension Word                          *
POWD    1     V86       Power Float(55)                              *
POWF    1     V85       Power Float(23)                              *
POWW    1     V83       Power Fixed(15)                              *
POWX    1     V84       Power Fixed(31)                              *
PREV    1     V88       Prevent (given Task)                         *
PROC    1     V18       Call procedure                               *
PUT     3     V27       Eröffnung einer Liste für  PUT               *
QSHC    1     MOVEQ ..,D2  Quick shiftcount                          *
RBRK    -     V42       Right Bracket in Format                      *
READ    3     V108      READ-Instruction                             *
REQU    -     TRAP =6   Request SEMA (adr. by A1)                    *
RELA    -     TRAP =7   Release SEMA (adr. by A1)                    *
RETN    -     TRAP =12  Return from procedure                        *
RFEN    -     V112      Remote-format end                            *
RLFO    1     V34       Remote-Left-Bracket in Format                *
ROLL    -     ROL.L D2,D0                                            *
ROUN    -     JSR -80(A6)   ROUND-function                           *
ROUN)   -     V143      Nicht für Neuentwicklung (ROUND)             *
RWND    1     V49  **   REWIND Dation and open                       *
SAVP    2     V57  **   Save a position in a file                     *
SEEK    2     V56  **   Seek a position in a file                     *
SHFL    2     V21       Shift 17 to 32 BIT Object                    *
SHFS    2     V20       Shift 1 TO 16 BIT Object                     *
SIGN    1     Macro     LEA obj,A1 + JSR -84(A6) SIGN-function       *
SIGN)   1     V144      Nicht für Neuentwicklung (SIGN)              *
SKFO    1     V47       SKIP-Format                                  *
SSHC    -     MOVE D0,D2   Store Shiftcount in D2                    *
STAD    1     V136      Nicht für Neuentwicklung (St. accu 8 by)*
STAL    1     MOVE.L D0,xx(A5)                                       *
STSC    -     V100      Start schedule-definition                    *
SUBD    1     Macro     LEA obj,A1  +  JSR  -24(A6)                  *
(SUBD)  1     V5        Nicht für Neuentwicklung (Sub Float 55)      *
SUBF    1     Macro     LEA obj,A1  +  JSR    -8(A6)                 *
(SUBF)  1     V4        Nicht für Neuentwicklung (Sub Float 23)      *
SUBW    1     SUB.W     Native Subtract Word                         *
SUBX    1     SUB.L     Native Subtract Long                         *
SUSE    1     V64       Suspend external (given Task)                *
SUSP    -     TRAP $A028   Suspend self                              *
SU20    -     SUB D2,D0   (In FOR ... REPEAT)                        *
SWAP    -     SWAP D0                                                *
SWLT    -     Macro     SLT D0  + EXT D0                             *
SYNC    1     V50  **   Synchronize Dation                           *
TACT    2     V97       (Timed)..activate with priority              *
TCON    1     V94       (Timed) ..Continue                           *
TERM    1     V87       Terminate (given Task)                       *
TFOR    2     V127      T-format                                     *
TRES    -     V99       (Timed) .. Resume (own Task)                 *
TSTW    -     TST.W D0                                               *
TSTL    -     TST.L D0                                               *
UNIL    1     V104      UNTIL (clock) schedule param. set            *
VARD    1     V13       Param. XFER: Variable Float 55               *
VARF    1     V12       Param. XFER: Variable Float 23               *
VARW    1     V10       Param. XFER: Variable Fixed 15               *
VARX    1     V11       Param. XFER: Variable Fixed 31               *
```

WACT	2	V96	(When)..activate with priority	*
WCON	1	V93	(When)... Continue	*
WHEV	1	V106	WHEN (event) schedule param. set	*
WRES	-	V98	(When) .. Resume (own Task)	*
WRIT	3	V107	WRITE-Instruction	*
XFOR	1	V33	X-Format	*

Vor dem Studium der Codierung eigener E/A-Treiber sollte man sich mit der typischen Maschinencode-sequenz einer Ein- oder Ausgabe vertraut machen. Dazu betrachten wir den Fall einer E/A über den ACIA/SCC mit der Warteschlangennummer 2. Es empfiehlt sich, das Assemblerprogramm durch den folgenden universellen I/O-Vorspann lesbarer zu machen. Man lese dazu auch unter E-III die Beschreibung des Communication Elements (CE).

```
*        Define Systemtraps:
FETCE    OPD    $4E48    Fetch communication-element (CE) Systrap
IOWA     OPD    $A00A    I/O wait function
RELCE    OPD    $4E49    Release the CE
XIO      OPD    $4E4A    Xfer communication-element for I/O
*        Define relative displacements in communication-element:
BUADR    EQU    $20      4 byte long buffer-address (from FETCE)
FNAME    EQU    $2A      File-name + type + drive
LDNIO    EQU    $27      Logical dation number (=Queue-number)
MODE     EQU    $28      Mode-byte of communication-element
RECLEN   EQU    $24      Record length (16 bit)
STATIO   EQU    $26      Statusbyte of communication-element
*        Symbolic masks and bit-positions:
MODMOU   EQU    $40      Mode-mask  for  Output
MODMCR   EQU    $20      Mode-mask  'carriagereturn ends record'
MODMWA   EQU    $80      Mode-mask 'wait for completion'
MODM..                   Other masks when used
STABRE   EQU    1        'Verschrottungsbit'
```

FALL A: Es soll eine Ausgabe gestartet werden und während
------ des Transfers noch etwas anderes gemacht werden. Das CE soll anschließend für einen weiteren E/A-Vorgang erneut benutzt werden. Textlänge sei 50 characters. Wir codieren:

```
.....    .....          Arbitrary code before write sequence
MOVEQ   =50,D1   Communication-elem. with 50 char. infolen.
FETCE            Get space from RTOS-UH, A1 is loaded
MOVE.B =MODMOU+$0,MODE(A1) Xfer-mode: out,no wait,by cnt.
MOVE.B =2,LDNIO(A1)   Queuenumber = 2  (Dation=ACIA2)
CLR.B  FNAME+11(A1)   Ax (not Bx or Cx)
MOVEA.L BUADR(A1),A2  pointer 1st character in buffer
.....
MOVE.B ...,(A2)+      put info into communication elem.
.....
MOVE    =50,RECLEN(A1)  all 50 characters to write
XIO              Make the output
....             other activities while output is running
....             A1 must be saved and reloaded !!
IOWA             Wait for completion of I/O with A1-com.el
....             from here on:  A1-CE may be used again
RELCE            Release the CE in A1. A1 is invalid now
```

FALL B.

Über B2 soll ein Stück Text gelesen werden und dabei noch Programmaktivität während des Lesens stattfinden. Außerdem soll kein Echo erzeugt werden, der Zugang zum Bedieninterface ist zu verriegeln, damit die Zeichen $01 etc. gelesen werden können. Maximale Textlänge sei 40. und beim ersten auftretenden Carriagereturn soll der Xfer beendet werden. (Das Zeichen CR steht dann als letztes im Puffer). Wir codieren zusätzlich zum Universalvorspann:

```
MODMCR  EQU     $20      End Xfer with 1st carriagereturn.
MODMNE  EQU     $02      'No echo' (Only ACIA/SCC)
MODMSC  EQU     $04      Suppress command-interf. (only ACIA/SCC)
.....
MOVEQ  =40,D1  D1.L  = length of info in CE
FETCE          load A1 by RTOS-UH
MOVE.B =2,LDNIO(A1)  queue-number is 2
MOVE.B =MODMCR+MODMSC+MODMNE+$0,MODE(A1) Xfer mode setup
MOVE   =40,RECLEN(A1)  Max. number of char's
MOVE.B =2,FNAME+11(A1)   'B' (buffered) instead of 'A'
XIO            Make the Xfer
....           Arbitary-code, command-if remains blocked.
IOWA           Wait until record ready (CR or 40 char's).
*     Die Daten liegen nun bereit. In RECLEN(A1) ist die Anzahl
*     Bytes zu finden. Auf BUADR(A1) die 4 bytes lange  Adresse
*     des ersten Zeichens. Nach Auswertung der Daten:

 RELCE          Release CE. A1 invalid from now on!

*     Anmerkungen zum hier demonstrierten Beispiel:
*     Damit das Bedieninterface  wieder 'befreit' wird, muß ein
*     ACIA2-XIO   ohne  den 'suppress command-if'  mode  folgen
*     oder der Terminalbediener drückt die 'break'-taste.
```

FALL C.

'Ausgeben und vergessen' eines Konstantenstring. Zusätzlich zum Universalvorspann codieren wir nun:

```
.....
CLR.L  D1       Length of textbuffer = 0
FETCE           Fetch a CE
MOVE.L =textadr,BUADR(A1)   Start adr of text to write
MOVE.B =MODMOU+MODMCR+$0,MODE(A1) Output until CR
BSET   =STABRE,STATIO(A1)   'Verschrottungsbit'
MOVE   =100,RECLEN(A1)      Max. number of char if no CR
MOVE.B =2,LDNIO(A1)         queue-number
CLR.B  FNAME+11(A1)         Set to Ax
XIO                         A1 no longer valid!

*    Die Ausgabe läuft,  das CE ist  für diese  Task nun nicht
*    mehr erreichbar, da  nicht  festzustellen ist, ob es noch
*    existiert
```

Im Betriebssystem RTOS-UH sollen alle Peripheriegeräte, bei denen Wartezeiten einmalig > 0.6 ms bzw. wiederholt > 80 us anfallen, durch sog. 'Systemtasks' (Treibertasks) betreut werden. Dabei wird durch die Usertask mit dem 'XIO'-Trap eine prioritätentengeordnete Warteschlange angelegt. Anschliessend aktiviert XIO über eine 'LDN'-Tabelle die Treibertask.

Für Peripherie, die dem Prozessor keine verwendbaren Zeitreste läßt, wird diese Konstruktion nicht benötigt. So wird z.B. über den VME- oder PIA-Bus mit SEND und TAKE direkt und ohne Taskwechsel kommuniziert.

Wenn Sie Ihr System mit weiteren Peripheriegeräten ausbauen, und dafür keine der im RTOS-UH vorhandenen Treibertasks benutzt werden kann - z.B. neues V24-Port , dann ist das System in der hier angegebenen Weise zu erweitern. Die Aufgabe, die hierbei zu erledigen ist, gliedert sich in drei Phasen:

1. Festlegung einer 'LDN' für das neue Peripheriegerät.

   Es gibt zwei verschiedene Ausgangspositionen:

   a) System soll fortan dauerhaft die neue Station enthalten: Boot/ROM-Erweiterung. Zweckmäßigerweise wird die nächste freie LDN belegt. Geht man nach oben darüberhinaus, so wird vom Nukleus beim Autolinking die LDN-Lücke für nachladbare I/O-Tasks verwaltungstechnisch aufbereitet. (Tabellenplatz) Die Station muß mit einem üblichen alphanumerischen Namen versehen werden, der die Verwendung durch das Bedieninterface ermöglicht. Dazu ergänzen wir eine 'Scheibe' vom Typ No. 9. Die Stationseigenschaften werden mit einer Scheibe des Typs 10 vorbesetzt. (Analog zum 'SD'-Bedienbefehl). Die notwendigen Informationen sind unter E-VII 'Scheibenstruktur' zu finden.

   b) System soll nur vorübergehend um die neue Datenstation erweitert werden: In diesem Falle müssen Sie eine LDN aus einer der vorhandenen Lücken wählen. Die Station ist bei dieser Form der Erweiterung allerdings immer nur als 'Lx' vom Bedieninterface errreichbar. Die Stationseigenschaften können nur mit Hilfe des SD-Bedienbefehles besetzt werden. Außerdem kann bei dieser Form der Erweiterung nur die Laufwerksnummer 0 angesprochen werden.

2. Codierung einer I/O-Task für die gewählte Warteschlange.

   Der Taskname ist frei wählbar. Allerdings sollte bei einer Boot/ROM-Erweiterung dieser mit '#' beginnen, um die Task vor dem 'UNLOAD' zu schützen. Beim Entladen einer I/O-Task ist in jedem Fall besondere Vorsorge zu treffen, sodaß der Eintrag in der LDN-TID-Tabelle des Nukleus wieder gelöscht wird.

Damit der von irgendwo exekutierte 'XIO'-Trap sein Communication-element (siehe dazu Abschnitt E-III) weiterleiten kann, benötigt er eine Zuordnungstabelle, die die LDN in den Task-Identifier (TID) umwandeln kann. Beim Autolinking baut der Nukleus diese Tabelle auf. (Siehe Scheibe 1). Bei den nachmontierten Tasks müssen Sie (oder jemand anders) diesen Eintrag selbst erledigen. Nicht belegte Tabellenplätze sind durch den Eintrag einer (4-Byte) 0 markiert.

```
*       Task-head mit Namen,Priorität etc. (s. D-III)
* Oder: Scheibencode der Scheibe no. 1 (siehe E-VII)
*       Hohe Priorität wird empfohlen, z.B. -1 oder 0
*
*       System-traps needed here:                          *
QDPC    OPD     $4E43       Dispatcher-caller              *
OFF     OPD     $4E4F       All interrupts 'off'           *
RELCE   OPD     $4E49       Release Comm.element           *
TOQ     OPD     $4E4D       Take of queue                  *
TERMI   OPD     $4E41       Terminate (self)               *
*
*       Other displacements inside RTOS                    *
BLOCK   EQU     $24         Block-byte of a Task           *
BLKBSU  EQU     4           Suspend-bit-no. in BLOCK       *
TID     EQU     $802        Actual running Task ident.*
*
*       Interrupt-buffer                                   *
*       a)  Platz über Scheiben 2 ... 8 (s. E-VII)         *
*       b)  Platz im RAM freihalten:                       *
IRLINK  DC      0,0,0,0,..0  as used by ir-process         *
*
START   MOVEQ   =ldn,D1     Eigene LDN nach D1             *
nur b)  MOVEA.L $852,A1     Tab-pointer LDN-TID            *
nur b)  MOVE.L  TID,ldn*4(A1)  Montieren der I/O-Task*
        TOQ                 Take off queue                 *
        TERMI               Wird exekutiert, wenn leer*
nur a)  MOVEA.L $832...$84A,A0   Buffer-pointer IDPx *
nur a)  LEA     OFFS(A0),A0  benutzten Ber. auslassen*
nur b)  LEA     IRLINK,A0   access to ir-link-block        *
        MOVE.L  A1,(A0)+        Comm.element for ir        *
        MOVEA.L TID,A2      Link to Task-identifier        *
        MOVE.L  A2,(A0)+    fuer ir-process                *
        CLR     (A0)+       Character-index reset          *
        ....                hier möglicherweise andere*
        ....                Interruptunkritische Op.       *
*                                                          *
*       Beginn der unteilbaren Sequenz:                    *
*
        OFF                 Alle Unterbrechungen sperr*
        ....    (coupler)   Interrupt-process mit Para-
        ....                metern  versorgen, Hardware
        ....        ''      'scharf' machen, IR-PC         *
        ....                versorgen etc.                 *
nur b)  MOVE.L  =IRENT,IVEC Interr. vect. anschließen

        ....                !!  privilegierter Mode !!*
```

```
        BSET    =BLKBSU,BLOCK(A2) Task suspendieren
        ANDI    =$D8FF,SR    Interrupts einschalten    *
        QDPC                 Dispatcher starten.       *
*                                                      *
*       Die Task schläft nun bis der IR-Process sie    *
*       wieder freischaltet.                           *
*                                                      *
        RELCE                Freigabe des Comm.elem.   *
        BRA     START        nächstes El. aus Schlange *
```

3. Codierung des Interruptprozesses.

Um Fehler innerhalb von Interruptanwortroutinen (wrong opcode, wrong address etc.) auf eine interruptspezifische Art in einen geordneten Rückfall ableiten zu können, gibt es die Zelle IID ($7FE) = 'Interruptidentifier'. Der Nukleus erkennt an IID, über welchen Vektor der aktuelle Prozeß gegangen ist und ermittelt damit die Ansprungadresse für den Fehlerfall. Dazu dient ein vor der Einsprungstelle liegendes relativiertes 16-Bit Wort. Dieses Wort enthält die Differenz zwischen Fehlereinsprungort und regulärer Einsprungadresse. Die 'Malfunction'-routine muß dafür sorgen, daß z.B. alle Register restauriert werden und ggf. der Interruptverursacher (Coupler etc.) ersatzweise befriedigt wird.

Klar, daß die Zelle IID zunächst gerettet werden muß, denn der Interruptprozeß könnte ja einen anderen unterbrochen haben, dessen Vektorlink am Ende wiederhergestellt werden muß.

Typische Aufgabe des Interruptprozesses kann z.B. die Behandlung des Dateninterrupts sein. Nach Abschluss der Datenübertragung muß die suspendierte Task wieder freigegeben werden. Damit man auf Interruptebene zu deren Blockbyte Zugriff erhält, wurde bereits auf Taskebene vor der Selbstsuspendierung die Adresse des Blockbytes im Interruptdatenblock abgelegt.

-> Die Benutzung von Traps und BSR-Befehlen ist wegen der damit verbundenen Paralysierung des 'Rückfallmechanismus' gefährlich!! BSR/RTS kann durch LEA ...,Ax und JMP (Ax), eine obendrein schnellere Lösung, ersetzt werden oder aber man rettet nach dem IID- und Registersave das System-A7 in ein eigenes Register, um es bei Malfunction als erstes wieder auf den korrekten Wert zu bringen.

-> Es dürfen allenfalls 'harmlose' Traps ohne Änderung zentraler Systemdaten benutzt werden, etwa CSA, QSA etc.

-> Jedes Verlassen der Interruptroutine darf nur über den Dispatcherausgang erfolgen, damit während der IR-prozedur aufgelaufene Taskzustandsänderungen nicht 'verschlafen' werden!! Verwenden Sie also nie den RTE-Befehl außer in der unten beschriebenen Umgebung.

Die Interruptantwortroutine liegt typischwerweise im gleichen Programm wie die Grundebenentask, bedenken Sie aber, daß zum Zeitpunkt des Interrupteintrittes irgendein völlig fremder Prozeß die Prozessorregister etc. besitzt. Sie müssen größte Aufmerksamkeit darauf verwenden, daß der unterbrochene Prozess korrekt fortgesetzt werden kann!!

Schematischer Aufbau der IR-Antwortroutine:

```
IID      EQU       $7FE        Interrupt-identifier      *
DPC      EQU       $800        Dispatcher-call flag      *
*
         DC   IRMAL-IRENT      Rückfall-link             *
IRENT    MOVE      IID,-(A7) Rette den alten Ir-Ident.*
         MOVE      =IVEC,IID   z.B. $200 nach IID      *
         MOVEM.L   D0...,-(A7)     Save Registers used *
nur b)   LEA       IRLINK,A0 Parameter-feld anschl.    *
nur a)   MOVEA.L   $832...$84A,A0      IDPx laden      *
nur a)   LEA       OFFS(A0),A0    fremden Bereich skip*
         .......               Von der Task-grundebene  *
         .......               angelegte Daten, z.B. die*
         .......               Adresse des CE, sind ueber
         .......               xx(A0) erreichbar        *
         MOVEA.L   (A0),A1       access to comm.elem.   *
         MOVE.B    ...,...     Daten ueber Coupler      *
         .......                                        *
         CMP       ...,...     Test ob Transfer fertig  *
         BEQ       lbl2        Sprung wenn fertig       *
         .....                                          *
exit     MOVEM.L   (A7)+,D0...   Register rückladen     *
         MOVE      (A7)+,IID   Alten Inter. Id. holen   *
         ORI       =$700,SR    All IR off               *
         TST       DPC         Dispatcher-call?         *
         BMI.S     lbl1        B:No process switching   *
         RTE                   normal exit              *
lbl1     JMP       $906        Call Dispatcher + Exit   *
*                                                       *
lbl2     MOVEA.L    4(A0),A0      access Task-id        *
         BCLR      =BLKBSU,BLOCK(A0) continue task      *
         SUBQ      =1,DPC      MARK DSP-CALL            *
         MOVEM.L   (A7)+,D0...  Reload registers        *
         JMP       $906        Call dispatcher          *
*                                                       *
IRMAL    ....                  Controller totschlagen   *
         BRA       exit        Exit by Disp. test       *
```

4. Andere Konstruktion ohne Interrupts.

Liefert der Coupler keine Interrupts, so kann auf der Ebene der Task selbst das Communicationelement bearbeitet werden. Um Verluste durch Abfrageschleifen zu vermeiden, bediene man sich einer 'durchlöcherten' Schleife, etwa, indem die Task alle 4 msec nachschaut und sich bei fehlender Bereitschaft des Couplers erneut selbst suspendiert für 4 msec. Bei hohen Datenraten geht es natürlich auch ganz ohne Suspendierung der I/O-Task.

5. Anmerkungen zum Beispiel.

Es handelt sich um ein einfaches Programm, welches einen normalen CE-Transfer zu einem ACIA-Baustein gestattet. Zur Demonstration wurde auch die typische ausgereizte Sequenz zur Erkennung von CR, LF oder EOT angegeben.

Das Maschinenprogramm braucht nur noch zugeladen zu werden. Die Task ist einmal zu aktivieren, damit der Tabelleneintrag erfolgen kann. (Wegen leerer Schlange keine Aktion). Anschließend ist über SD L5 xx das Port zu parametrieren. Bevor die Task mit UNLOAD entfernt wird, muß entweder über eine Hilfstask oder mit Hilfe des SM-Befehles (Vorsicht! sorgfältig rechnen) wieder eine Null an Stelle des TID eingesetzt werden. (4 Byte Null)

```
*************************************************
*                                               *
*       Demonstrationsprogramm   'eigene  E/A'  *
*       HIER: Nachmontierte Version             *
*                                               *
*       Taskname: Queue5, LDN=5                 *
LDN     EQU     5               For assembler   *
*-----------------------------------------------*
*       TASK-HEAD FOR RTOS-UH:                  *
*                                               *
        DC.B    0,0,0,0,0,0,0,0,0,1,'Queue5'    *
        DC      0               Priorität       *
        DC.L    120             Minimum WSP     *
        DC.L    0,0             for System      *
        DC      0               also priority   *
        DC.L    START           Start-PC        *
        DC.L    0,0,0,0,0,0,0,0,0,0             *
        DC.L    0,0,0,0,0,0     at least:64 Bytes 0 *
*-----------------------------------------------*
*       Coupler- and interrupt- addresses:      *
*       *** Depending upon actual hardware ***  *
*           ----------------------------        *
ACST    EQU     $50041          Statusreg ACIA  *
ACDT    EQU     $50043          Datareg ACIA    *
IVEC    EQU     $78             Interrupt-link  *
*-----------------------------------------------*
*       System traps:                           *
QDPC    OPD     $4E43           Dispatcher-caller *
OFF     OPD     $4E4F           All interrupts off *
RELCE   OPD     $4E49           Release comm.elem. *
TOQ     OPD     $4E4D           Take off queue  *
TERMI   OPD     $4E41           Terminate self  *
```

```
*-------------------------------------------- *
*       Other displacements                    *
BLOCK   EQU     $24             Block-byte     *
BLKBSU  EQU     4               Suspend-bit number  *
IID     EQU     $7FE            Iinterrupt-identif. *
DPC     EQU     $800            Dispatcher-call Flg *
TID     EQU     $802            Actual running Task *
LDNTID  EQU     $852            ldn->TID Table     *
DISEX   EQU     $906            Dispatcher-Exit    *
*       In communication-element:              *
RECLEN  EQU     $24             Record-length      *
MODE    EQU     $28             Mode-byte          *
 MODMCR: EQU    $20             Stop on carriage rtn*
 MODMLF: EQU    $10             Stop on Line-feed  *
 MODMEO: EQU    $08             Stop on EOT        *
BUADR   EQU     $20             Adress of text     *
*-------------------------------------------- *
IRLINK DC.L     0               Actual Text-address *
       DC.L     0               Task-ID of this Task*
       DC       0               Length-control-word *
*-------------------------------------------- *
*                                              *
*       T A S K  -  C O D E                    *
*                                              *
*                                              *
START  MOVEA.L LDNTID,A0       Table-address   *
       MOVE.L TID,LDN*4(A0)    Nachmontage LDN *
       MOVEQ  =LDN,D1          Own ldn         *
       TOQ                     Inspect the queue *
       TERMI                   Executed if empty *
       LEA    IRLINK,A0        For rapid access *
       MOVE.L BUADR(A1),(A0)+  Start adr of Text *
       MOVEA.L  TID,A2         Task Id for access *
       MOVE.L A2,(A0)+         save for ir-process *
*                                              *
*       Determine number of chars to transmit  *
*                                              *
       MOVE   RECLEN(A1),D2    Assumed length  *
       MOVEQ  =MODMCR+MODMLF+MODMEO,D6  Testmask *
       AND.B  MODE(A1),D6      Quick check     *
       BEQ.S  A06              b: no mode specified*
       MOVEA.L BUADR(A1),A2    Text-org        *
       CLR    D3               Reset Record-length *
A02    CMP    D2,D3            Reclength test  *
       BGE.S  A06              b:all done      *
       ADDQ   =1,D3            Move counter    *
       MOVE.B (A2)+,D0         Inspect the byte *
       MOVEQ  =MODMCR,D7       Test-mask       *
       SUB.B  =$0D,D0          Test for Carr. rtn *
       BGT.S  A02              b: not cr,lf or eot *
       BEQ.S  A04              b:is cr         *
       MOVEQ  =MODMLF,D7       Testmask        *
       ADDQ.B =$0D-$0A,D0      Test for LF     *
       BEQ.S  A04              b:is LF         *
       ADDQ.B =$0A-$04,D0      Test for EOT    *
       BNE.S  A02              b:not eot       *
       MOVEQ  =MODMEO,D7       Testmask        *
A04    AND.B  D6,D7            Mode-match?     *
       BEQ.S  A02              b:no match      *
       MOVE   D3,D2            result length   *
```

```
A06     MOVE    D2,(A0)+        Store length          *

        LEA     IRENT,A3        Ir-entry address      *
*
        OFF                     Disable interrupts    *
        MOVE.B =$35,ACST        New coupler status    *
        MOVE.L A3,IVEC          Ir-vector connection*
        BSET    =BLKBSU,BLOCK(A2)  suspend the task*
        ANDI    =$D8FF,SR       Interrupts 'on'       *
        QDPC                    Call dispatcher       *
*       Now the task is suspended for last interr.*
        RELCE                   Release Caller        *
        BRA     START           Repeat queue-op.      *
*                                                     *
*-----------------------------------------------------*
*                                                     *
*       I N T E R R U P T - P R O C E S S             *
*                                                     *
*                                                     *
        DC      IRMAL-IRENT     Malfunction           *
IRENT   MOVE    IID,-(A7)       Save old Interr. ID *
        MOVE    =IVEC,IID       For Malfunc process *
        MOVEM.L A0/A1,-(A7)       Save registers      *
        LEA     IRLINK,A0       Link to parameters    *
        SUBQ    =1,8(A0)        Counter control       *
        BMI.S   IRCOD4          b:end of transm.      *
        MOVEA.L (A0),A1         Buffer-address        *
        MOVE.B  (A1)+,ACDT      Send data to periph.*
        MOVE.L  A1,(A0)         Restore new pointer *
IRCOD0  MOVEM.L  (A7)+,A0/A1          reload reg's    *
        MOVE    (A7)+,IID       Recall old Inter.ID *
IRCOD1  ORI     =$700,SR        All ir off            *
        TST     DPC             Is there any call ? *
        BMI.S   IRCOD2          B:Yes there is        *
        RTE                     Exit of Ir-process    *
IRCOD2  JMP     DISEX           Exit by dispatcher    *
*                                                     *
*       End of transmission                           *
IRCOD4  MOVE.B =$15,ACST        Switch coupler off    *
        MOVEA.L 4(A0),A0        Access task-id        *
        BCLR    =BLKBSU,BLOCK(A0)  Continue           *
        SUBQ    =1,DPC          Flag dispatcher-call*
        MOVEM.L (A7)+,A0/A1          reload reg's     *
        MOVE    (A7)+,IID       Recall old Inter.ID *
        JMP     DISEX           Exit by dispatcher    *
*                                                     *
*       Malfunction recovery-exit                     *
IRMAL   BRA.S   IRCOD0                                *
*                                                     *
        END
```

This page intentionally left blank.

RTOS-UH Autolink -- die Systemkonfigurierung

Das Betriebssystem besteht aus einer Vielzahl kleiner Module, die vollkommen lageunabhängig sind. Die gegenseitigen Verbindungen zwischen den Modulen werden erst nach Einschalten des Systemes im Grundmodul, dem 'Nukleus', hergestellt. Dazu baut der Nukleus im unteren Speicherbereich ab $800 eine Vielzahl von Tabellen auf, in denen die Querbezüge ihren Niederschlag finden. Die Querbezüge werden durch Abtasten ('Scanning') eines vereinbarten E-PROM (oder RAM-) Bereiches aufgespürt. Signalmarke ist dabei die auf gerader Adresse beginnende Bytesequenz $AEB1BF95, gefolgt von einem Wort mit einem ungeraden Vielfachen der Primzahl 37. Wir sprechen von der 'Scheibe x', wenn dieses Wort genau den Wert (x*2+1)*37 besitzt.

Die Signalmarke ist so gewählt, daß sie im normalen Assemblercode sowie im VCP-code nicht vorkommen kann. Man hüte sich aber, ungenügend vorbesetzte Speicherbereiche abtasten zu lassen. Auch besteht eine gewisse Gefahr, wenn grosse Datenfelder im Betriebssystembereich abgelegt werden sollen.

Das 'Zusammenbauen' eines Betriebssystemes besteht nun einfach in einer möglichst lückenlosen Hintereinanderreihung der einzelnen Module ('Scheiben') und dem Besetzen des Scanbereiches in den dafür vorgesehenen Zellen des Nukleus.

!! Das System startet auf der Adr. Nukleus+$18 im Supervisormode.

--> Der Scanbereich muß immer mit der Anfangsadresse des Nukleus beginnen. Er kann absolut oder relativ zur Nukleusanfangsadresse definiert werden. Dabei sind zwei getrennte Bereiche möglich. Dies geschieht durch Eintragung von 4 Byte-Adressen auf den Zellen Nukleus+$20.

```
Bsp:  Nukleus+$20      DC.L   $00000001,$0000FFFF  (relativ)
            +$28       DC.L   $F60000,$F7FFFE      (absolut)
            +$30       DC.L   0                    Stop-marke
```

Gesetztes LS-Bit heißt 'relativ zum Nukleus'. In der ersten Sektion werden also 64 kB, in der zweiten 128 kB abgetastet. Fehlt der zweite Bereich, so ist Nukleus+$28 auf $00000000 zu setzen.

--> Denken Sie also bitte daran, daß bei einer Erweiterung des Systemes ggf. der Scanbereich im Nukleus angepasst werden muß. Dabei genügt es, wenn die letzte Signalmarke noch inhalb des Bereiches liegt, der Code darf also darüberhinausragen.

--> Beim Scanning darf kein Bus-Error auftreten, sonst läuft das System nicht an.

Das kleinste denkbare System besteht nur aus dem Nukleus. Allerdings leistet es nichts, außer daß die Task #IDLE dauernd läuft. Durch Hinzufügen einer oder mehrer Scheiben wird daraus entweder ein kleines reines Laufsystem oder ein komfortables Entwicklungssystem, je nach dem beabsichtigten Einsatz.

RTOS-UH Autolink -- Modifikation eines Systems

Ein normal aufgebautes Entwicklungssystem unfaßt folgende Module:

NUKLEUS	Betriebssystemkern, enthält Speicherverwaltung, Taskmanagement, stellt Systemtraps zur Verfügung
ERROR	Fehlerauswerter, ist für korrekte Auswertung und Ausgabe von im NUK erkannten Fehlsituationen erforderlich
SHELL	Kommandointerpreter
EDFM	Edit-File-Manager, verwaltet die unter den Datenstationen ED und EDB angesprochenen Dateien und enthält die Datenstationen VI/VO
IMP	Implementierungsscheibe, enthält Treiber für serielle Schnittstellen und weitere Hardware-spezifische Teile
ED	Editor
LOAD	Lader
MATH	Mathe-Pack, enthält nicht elementare Gleitkommaarithmetik (Sinus, Logarithmus etc.)
HYP	Hyperprozessor, bietet Laufzeitunterstützung für PEARL-Programme
VCP	Virtual-Code-Processor, erforderlich für Assembler und Compiler
UHFM	File-Manager für Massenspeicher (Floppy, Winchester)
ASSEM	Assembler
COMP	Compiler

Sämtliche dieser Module sind aus Scheiben gemäß der in diesem Kapitel folgenden Beschreibung aufgebaut.
Zusätzlich zu den Modulen umfaßt eine Implementation die Integration eines Schutzmechanismus, der Veränderungen des Codes im Eprom verhindert.

Die Implementation eines Systems umfaßt die Erstellung des (rechnerspezifischen) IMP-Moduls, die Zusammenstellung der einzelnen Module in geeigneter Reihenfolge sowie die Festlegung des zu schützenden Code-Bereiches. Es sind hierbei verschiedene Systemkonfigurationen mit unterschiedlichem Leistungsumfang denkbar:

- Minimalsystem
  umfaßt nur das Modul NUK. Das Betriebssystem läuft vollständig, alle Traps sind angeschlossen, Schnittstellen sind nicht ansprechbar.
  Ein derartiges System hat praktisch keinen sinnvollen Einsatz, da Kommandointerpreter und Schnittstellentreiber fehlen. Es kann jedoch Programme, die keine Unterstützung durch EDFM, MATH und HYP benötigen, aus dem EPROM heraus exekutieren.
- Laufzeitsystem für Assemblerprogramme
  umfaßt NUK, ERROR, SHELL und IMP. Das System kann nach außen kommunizieren, Kommandos empfangen und exekutieren.
  Das System leistet ähnliches wie das Minimalsystem; es ist jedoch von außen steuerbar.
- komplettes Laufzeitsystem
  ist erweitert um EDFM, MATH und HYP, ggf. UHFM. Das System kann sämtliche in PEARL oder Assembler erstellten Programme bearbeiten und ggf. den Massenspeicher verwalten. Da der Lader noch nicht vorhanden ist, können nur EPROM-residente Programme ausgeführt werden.
- potentielles Entwicklungssystem
  ist erweitert um LOAD und ED und VCP. Das System kann Programme

von Massenspeicher oder Schnittstellen laden und exekutieren. Damit kann das System beliebige Programme ausführen.
- komplettes Entwicklungssystem
  enthält zusätzlich Compiler und Assembler, entweder EPROM-resident oder von Massenspeicher nachgeladen. Hiermit ist der volle Leistungsumfang eines Standard-RTOS-UH/PEARL-Systems erreicht.

Bei einer regulären Implementierung eines Entwicklungssystems werden alle Module bis auf COMP und ASSEM vom Schutzbereich erfaßt, d. h. es ist möglich, das komplette Entwicklungssystem derart zu modifizieren, daß nur ein potentielles Entwicklungssystem EPROM-resident zur Verfügung steht. Je nach Hardware (s. Kap. F) kann dies durch Ersetzen der Compiler- und Assembler-EPROMs durch andere EPROMs (die z. B. Anwenderprogramme enthalten) oder durch Überschreiben von Compiler und Assembler in den EPROM's mit Anwenderprogrammen geschehen.

B E I S P I E L :

Betrachten wir als Beispiel ein System mit folgender Hardware-Konfiguration:

Prozessor:	68000
EPROM-Bestückung:	4x 27256, organisiert in zwei Bänken à 16 bit
EPROM-Adressen:	Bank1 : $F80000 - $F8FFFE Bank2 : $F90000 - $F9FFFE Bei einer derartigen Konfiguration enthält im Regelfall Bank1 das potentielle Entwicklungssystem (mit Schutzmechanismus) und Bank2 Compiler und Assembler (ungeschützt). Compiler und Assembler mögen die Adressen $0000 - $7ECC in Bank2 (relativ zu $F90000 belegen)

AUFGABE 1

Als einfachen Einstieg betrachten wir die Umdefinition der Eigenschaften einer Datenstation. Erforderlich hierzu ist Kenntnis der LDN dieser Station (s. Tabelle Kap. A-I-10 und ggf. Kap. F) sowie der Device-Parameter (s. Befehl SD, Kap. B-II).
Die zweite serielle Schnittstelle (A2:), die vom System standardmäßig als Datenschnittstelle mit den Device-Parametern $0B00 vorbesetzt wird, sei beim Kaltstart des Systems als Terminal-Schnittstelle mit den Device-Parametern $3300 zu initialisieren (Standard bei A1:).
Erforderlich hierzu ist die Erstellung einer 10-er Scheibe gemäß der Beschreibung am Ende dieses Kapitels.

Wir kodieren hierzu in Assembler

```
RORG    0               relativierende Assemblierung
DC      $AEB1           erstes Wort der Signalmarke
DC      $BF95           zweites Wort der      "
DC      (10*2+1)*37     Kennung Scheibe 10

DC      2               LDN der A2-Schnittstelle
```

```
        DC      $3300           gewünschte Device-Parameter
        DC      0               Endekennung dieser Scheibe

        END                     Ende Assemblierung
```

Wird dieser Text assembliert, so erhält man

a) in der Assemblerliste eine auf Adresse 0 beginnende Speicherbelegung $AEB1, $BF95, $0309, $0002, $3300, $0000

b) die hierzu gehörenden S-Records in der CO-Datei des Assemblers.

Wird der generierte Code in der EPROM-Bank2 ab Adresse $7ECC abgelegt, so übernimmt das System die gewünschte Device-Parametrierung beim Kaltstart.

AUFGABE 2

Als etwas aufwendigere Aufgabe betrachten wir die Einbindung eines PEARL-Programmes in das System zur Ausführung als Autostart-Task nach Kalt-und Warmstart des Systems. Gleichzeitig sollen Compiler und Assembler aus dem System entfernt werden, um den verfügbaren EPROM-Platz zu vergrößern. Voraussetzung hierfür ist das Vorhandensein des 'PROM'-Befehls.

a) nicht elegant, aber einfach
Ist das PEARL-Programm geschrieben und übersetzt, so laden wir es (möglichst mit Size-Angabe gemäß Compilerbilanz)an die Speicheradresse, auf der es im späteren Betrieb ausgeführt werden soll. Anschliessend kann mit der Anweisung

```
AUTOSTART  modulname,taskname
```

die gewünschte Task in eine Autostart-Task umgewandelt werden.
--- Bitte jetzt nicht 'ABORT' betätigen - die Task würde sonst als Autostart-Task sofort starten ---
Mit dem Befehl

```
PROM modulname*
```

können wir nun (s. Beschreibung Befehl PROM) S-Records erzeugen. Diese S-Records sollten vom EPROMMER ab Adresse $0 der Bank2 in einen EPROM-Satz programmiert werden. Werden die so programmierten EPROMs statt der Compiler- und Assembler-EPROMs eingesetzt, so startet das System sofort nach Kalt-oder Warmstart unsere Autostart-Task.
Unser PEARL-Programm wird nun allerdings nicht aus dem EPROM heraus exekutiert, sondern beim Kaltstart auf die Adresse kopiert, auf der wir es beim PROMmen geladen hatten.

b) eleganterer Weg
Nach einem ersten Compilerlauf merken wir uns die Größenangabe aus der Compilerbilanz ('$xxxx BYTES'), z. B. E78.
Für einen zweiten Compilerlauf fügen wir vor den Anfang des PEARL-Moduls die Zeile

```
SC=$E78,CODE=$F90000,VAR=$3000;
```

in den Quelltext ein. Der Compiler generiert nun Code, der aus dem EPROM heraus exekutiert werden kann (beginnend auf Adresse $F90000), lediglich der Modulvariablenbereich muß hierzu im RAM

vorhanden sein. Nach dem Laden (auf beliebige Adresse) können wir mit der Anweisung

AUTOSTART modulname,taskname

die gewünschte Task in eine Autostart-Task umwandeln.
Mit dem Befehl

O ED:SR;PROM modulname*

können wir nun (s. Beschreibung Befehl PROM) S-Records erzeugen. Im File ED.SR erscheinen zwei S0 .. S2 .. S9 Blöcke (z. B. Zeile 1 - 27 und Zeile 28 - 67), die wir mit der 'SC'-Option des COPY-Befehls in getrennte Dateien kopieren (zwei Durchgänge).

Die S-Records des zweiten Blocks müssen mit einem EPROMmer ab Adresse $0 der Bank2 (entspricht der physikalischen Adresse $F90000, s. 'CODE='-Angabe) in einen Satz EPROMs programmiert werden. Die S-Records des ersten Blocks können in einen beliebigen, ausreichend großen Freiraum des EPROM-Satzes gebrannt werden. Haben wir sichergestellt, daß unsere EPROM-Daten noch vom Scan-Bereich des Systems erfaßt werden (ggf. Nukleus+$20 aufwärts ändern), so können wir unsere EPROMs statt Compiler und Assembler einsetzen. Beim Kaltstart des Systems richtet der Nukleus nun im RAM

a) den Modulvariablenbereich, beginnend bei Adresse $3000 gemäß der 'VAR='-Angabe, und
b) Taskköpfe für alle Tasks unseres Moduls

ein. Unsere Autostart-Task wird nach der Warmstartpase gestartet.

AUFGABE 3

Als aufwendigsten Fall betrachten wir die Einbindung eines E/A-Treibers gemäß dem Beispiel aus Kap. E-VI.

Zunächst kodieren wir einen Scheibenkopf für eine 1-er Scheibe (Systemtask-Definition):

DC	$AEB1,$BF95,(1*2+1)*37	
DC.B	$01,$80	Normale Task als Betreuungstask
DC.B	'MYQUE '	Name der Task
DC	-1	sehr hohe Priorität
DC.L	$100	256 Byte Workspace nötig (je nach Bedarf, muß aber > $66)
DC.L	START-$	relativierte Startadresse
DC	ldn	LDN wie im Beispiel
DC	0	Endekennung für diese Scheibe

Da wir unsere Routine im EPROM ablegen wollen, können wir eventuellen Speicherbedarf der Interrupt-Routine nicht über DC-Anweisungen reservieren. Daher müssen wir beim Kaltstart einen Speicherbereich als Interrupt-Puffer anfordern, d. h. als Speicherbereich, der nur unserer Interrupt-Routine zur Verfügung steht. Wir entnehmen Kap. F die schon belegten Interrupt-Puffer und wählen einen der freien Interrupt-Puffer.In diesem Beispiel verwenden wir den IDP5.

Zur Anforderung kodieren wir daher eine 6-er Scheibe:

```
        DC      $AEB1,$BF95,(6*2+1)*37   6-Scheiben-Marke
        DC      100                      z. B. 100 Byte Puffer
```

Das System reserviert hierdurch 100 Byte Speicherplatz. Die Adresse des ersten Bytes dieses Speicherbereiches finden wir nach dem Kaltstart des System auf der Adresse $842.

Weiterhin soll unsere selbstdefinierte Datenstation auch unter einem Namen (Mnemo) vom Bedieninterface her erreichbar sein (von der PEARL-Ebene her ist unsere Datenstation nur über LD/ldn,drive/ erreichbar).
Wir kodieren daher weiterhin eine 9-er Scheibe:

```
        DC      $AEB1,$BF95,(9*2+1)*37   9-er Scheiben-Marke
        DC.B    'MQ'                     Mnemo
        DC.B    ldn+$80                  LDN und Textende-Markier-
                                         ung
        DC.B    0                        Laufwerksnummer, wir neh-
                                         men hier drive 0
        DC.B    0                        Ende der Scheibe
```

und haben hiermit den Mnemo-Anschluß installiert.

Wollen wir auch noch Eigenschaften der Datenstation vorbesetzen, so können wir dies mit einer 10-er Scheibe (s. Aufgabe 1) tun.

Sollte unsere Datenstation über einen Peripheriebaustein mit der Außenwelt kommunizieren, so muß dieser Baustein im Regelfall beim Warmstart (Abort, oder nach der Reset-Kaltstart-phase) initialisiert werden. Hierzu kodieren wir eine 15-er Scheibe:

```
        DC      $AEB1,$BF95,(15*2+1)*37
        .....                            Maschinenkode zur Initia-
                                         lisierung
        RTS                              Rücksprung
```

Der Nukleus springt diese Code-Sequenz im Supervisior-Mode an. Bei der Erstellung des Maschinencodes müssen wir die bei der Beschreibung der Scheibe gemachten Einschränkungen beachten.
Nun können wir mit der Programmierung der Datenstation gemäß dem Beispiel aus Kap. E-VI beginnen:

```
START   MOVEQ   =ldn,D1                  Maschinencode wie im
                                         ..... Beispiel
        .....
        RELCE                            BRASTART
```

Da der von unserer Datenstation angesprochene Peripheriebaustein Interrupts auslösen kann, müssen wir noch einen Interrupt-Antwortroutine erstellen. Hierzu kodieren wir zunächst eine 14-er Scheibe:

```
        DC      $AEB1,$BF95,(14*2+1)*37  Kennmarke
        DC      $64,IRENT-$-2            setzt den Autovektor für
                                         den Level-1-Interrupt auf
                                         unsere Interrupt-Routine
        DC      0                        Endemarke
```

wobei wir annehmen, daß unser Peripheriebaustein nur einen Level-

1-Autovektor-Interrupt erzeugen kann. Selbstverständlich müssen wir hierbei die Eigenschaften der Hardware kennen und wissen, welche Interruptvektoren uns zur Verfügung stehen.

Nun können wir den eigentlichen Interrupt-Prozeß wie im Beispiel aus Kap. E-VI kodieren:

```
        DC      IRMAL-IRENT             Rückfall-Link
IRENT   MOVE    IID,-(A7)               Rette den alten IR-Ident.
        .....                           Maschinencode wie im
                                        Beispiel, merke: Inter-
                                        rupt-puffer über $842
                                        erreichbar
        .....
IRMAL   .....                           wie im Beispiel
```

Damit ist der gesamte Code für unsere Datenstation erstellt. Wir brauchen nur noch zu assemblieren und können den erhaltenen Code ins EPROM programmieren.
Beachten wir, daß ggf. der Scan-Bereich des Nukleus angepaßt werden muß, um alle Scheibenköpfe zu erfassen, und haben wir bei der Programmierung darauf geachtet, daß unser Programm frei im Speicher verschieblich ist (keine absoluten Sprünge etc.), so sollte unsere Datenstation einsetzbar sein.

Signalmarke: $AEB1,$BF95,$006F (Scheibe 1)

Bytes hinter der Signalmarke:

```
      $0                  Type  (s.u.)
      $1                  Class (s.u.)
      $2  ...  $7         Name-info
      $8  ...  $9         Priority
      $A  ...  $D         Size of required Workspace, 4 bytes.
      $E  ...  $F         Start-PC minus location of this word.
     ($10 ...  $11)       Only if class is $80 or $01: 'LDN'.
      $10  (or $12)       Type  (next task)
      $11  (or $13)       Class (next task)
      ...........
      ...........         Start-PC relative  or LDN  of  last task.
      ...........         $0000  Stop-marker this slice.
```

Type: Ist das spätere 2. Byte des Typwortes im RTOS-UH.

```
      $01                 Normale Task.
      $81                 Residente Task.
      $41                 Normale Task mit Autostart.
      $C1                 Residente Task mit Autostart.
-->   andere              n i c h t   e r l a u b t , r e s e r v.
```

Class: Gibt an, welche Sonderform der Task vorliegt.

```
      $00                 Normale Task.
      $01                 Bedieninterface für LDN ($10 ..$11, s.o.)
      $02                 Die Errortask des Systemes.
      $80                 Eine I/o-queue-Betreuungstask für die
                          Warteschlange der LDN auf $10 .. $11.
                          MSB der LDN null oder 2. LDN (z.B. VI/VO)
-->   andere              n i c h t   e r l a u b t ,  r e s e r v .
```

--> Es darf im System nur eine Errortask existieren. Ein Austausch der standardmäßig vorhandenen Error-task ist nur bei sehr kleinen reinen Laufsystemen sinnvoll - etwa um eine LED im Fehlerfall aufleuchten zu lassen etc. Die Übergabedaten finden sich im ringförmigen Errorpuffer des Systemes.

--> Bei den I/O-Tasks wird der entsprechende Anschluß für den Trap XIO automatisch hergestellt. Zum Bekanntmachen des Stationsnamens und der Eigenschaften sind besondere Scheiben vorgesehen.

```
Bsp.          DC     $AEB1,$BF95,(1*2+1)*37     Signalmarke.
---           DC.B   $C1,0      Resident, Autostart, normal
              DC.B   '#SELFT'   6 bytes name of task
              DC     $7FF0      Extremely low priority.
              DC.L   $66        Workspace ($66 is minimum!!)
              DC     START-$    Start-PC rel.
              DC     $0         Stop-marke für diese Scheibe.
              ......
      START   MOVE   ....       Anfang der Systemtask.
```

name-info:

Entweder	genau 6 ASCII-Bytes (wie im Beispiel)
Oder	relativierter 4-Byte-Zeiger auf den Namensstring, der durch $FF beendet wird. In diesem Fall haben die restlichen 2 Bytes keine Bedeutung, müssen aber angelegt werden. Die Relativierung erfolgt über DC.L Textadr-$

Anmerkungen:

Alle residenten Systemtasks erhalten bereits beim Kalt- oder Warmstart ihren Taskworkspace im Bereich unterhalb des dynamisch verwalteten Speicherbereiches. Damit soll der Verwaltungsaufwand reduziert werden.

Tasks mit gesetztem Autostart-bit (wie im Bsp.) laufen beim Aufsetzen des Systemes unter Berücksichtigung der festgelegten Priorität sofort los. Auch I/O-Betreuungstasks dürfen ggf. vom Autostart-typ sein, obwohl eine etwaige Peripheriebaustein-Initialisierung in einer Warminitialisierungsscheibe besser aufgehoben ist.

Signalmarken: $AEB1,$BF95,(x*2+1)*37 x=2,3,4,5,6,7,8

Das System stellt einen Interruptpuffer für den Level (x-1) zur Verfügung des Systemprogrammierers. Dabei wird während des Autolinking für jeden Level nach der größten Anforderung gesucht. Nur dieser Maximalwert wird berücksichtigt. Auf den Zellen 'IDPy' legt das System einen 4 Byte Zeiger ab, der auf den Anfang des Puffers für den Level y zeigt.

Wort hinter der Signalmarke: 2 Byte lange Puffergröße.

```
Bsp:        DC   $AEB1,$BF95,(4*2+1)*37      Level 3
            DC   100                         100 Bytes
```

Falls nirgendwo im abgetasteten Bereich für den Level 3 eine größere Anforderung gefunden wird, so werden genau 100 Byte Puffer für den Level 3 bereitgestellt.

--> Die Verwaltung der Displacements muß der Systemprogrammierer selbst übernehmen. Insbesondere dürfen bei Einbau dieser Scheibe schon angeforderte Zellen nicht benutzt werden. In Zweifelsfällen kann man aus den IDP-Differenzen im Zielsystem ablesen, ab welchen Displacements Platz definiert werden kann. (Im obigen Bsp: Inhalt IDP4 - Inhalt IDP3 = Puffergr.3)

IDP1=$832, IDP2=$836, IDP3=$83A, IDP4=$83E, IDP5=$842, IDP6=$846, IDP7=$84A

Beispiel zur Verwendung des Interruptpuffers:

```
HILF1    EQU     30          Start-displacement
HILFXY   EQU     34          Irgendeine Variable
SIZE     EQU     38          Letztes Byte auf displacem. 37
* ...
         DC     $AEB1,$BF95,(6*2+1)*37    Level 5
         DC     SIZE
* ...
         DC        MALF-IRENTR       Rückfallmechanismus
IRENTR   MOVE      IID,-(A7)         Save old Ir-Id. ($7FE)
         MOVE      =Vectoradr,IID    Neuer Interrupt vect.
         MOVEM.L   A0-A3,-(A7)       angenommenes Beispiel.
         MOVEA.L   IDP5,A0           Basiszeiger
         .....
         MOVE.L    HILFXY(A0),...    typ. Zugriff.
```

---> Der Zeiger IDP1 wird im System zum Löschen des gesamten Interruptpuffers (bis zum Ende des IDP7) bei jedem Warm- oder Kaltstart benutzt, darf also nicht vom Programmierer 'verbogen' werden. Theoretisch würde ein einziger IDP für alle Interrupts genügen, die Aufteilung nach Levels ist nur zur Übersichtlichkeit gemacht. Wegen des Abort-clear eignen sich die Zellen auch sehr gut für 'private' Semaphorvariable.

Signalmarke: $AEB1,$BF95,(9*2+1)*37

Mit dieser Scheibe werden dem System ein oder mehrere Namen von neu definierten Datenstationen hinzugefügt. Das Bedieninterface kann mit Hilfe der Informationen dieser Scheibe aus einem Textstring dann die zugehörige Warteschlangennummer ('LDN') und die zugehörige Untergliederungsnummer ('DRIVE') ermitteln.

Aufbau hinter der Signalmarke:

```
$0 ... ?    Alphanumerischer Textstring beginnend mit Buchstaben.
$x ..       1 Byte 'LDN' mit aufgeodertem Bit $80. (Dient gleich-
            zeitig als Endemarke für den Textstring).
$x+1        1 Byte 'DRIVE', z.B. Laufwerksnummer.
.............................................................
$x+2        Nächster Textstring wie oben etc.
$y          1 Byte 'LDN' plus $80
$y+1        1 Byte 'DRIVE'.
.............................................................
.....       Letzter Block mit Textstring, ldn,drive.
.............................................................
$00         Stopmarke, Ende dieser Scheibe.
```

Beispiel: Sie wollen die Station PLOTTER für die LDN 5 einrichten, Laufwerksnummer sei 0 bzw. don't care.

```
DC     $AEB1,$BF95,(9*2+1)*37      Signalmarke
DC.B   'PLOTTER',$80+5,$00         Name, ldn, drive
DC.B   0                           Ende der Scheibe.
```

--> Damit die Station wirklich benutzt werden kann, müssen Sie noch mit Hilfe der Scheibe 1 eine zur gewählten LDN passende I/O-task bereitstellen. (s. Scheibe 1)

--> Wählen Sie die LDN dabei nicht größer als nötig. Der Nukleus sucht im abgetasteten Speicher nämlich nach der höchsten LDN aller 1-er Scheiben (Nicht der 9-er). Diese bestimmt den freigehaltenen Platz in der Tabelle 'LDN to TID' (Zeiger SIOLDT = $852, 4 Byte pro LDN).

--> Man kann allerdings auch gezielt Lücken bei den LDN lassen. Diese Tabellenlücken können durch Anschluß von nachladbaren I/O-Tasks (im verwalteten RAM) sinnvoll genutzt werden. So wird die I/O-Task 'nachmontiert':

```
        ....
START   MOVEA.L  $852,A1        Tab-Zeiger
        MOVE.L   $802,5*4(A1)   eigene TID
        MOVEQ    =5,D1          eigene LDN
        TOQ                   aus Schlange
        TERMI                 leer: Ende
        ...
```

Task einmal von Hand starten, ist dann betriebsbereit.

Signalmarke: $AEB1,$BF95,(10*2+1)*37

Die mit den Befehlen SD und DD ('Set Device parameter' und 'Display device parameter') zugänglichen Geräteeigenschaften können mit Hilfe dieser Scheibe vorbesetzt werden. Beziehen sich mehrere solcher Scheiben auf das gleiche Gerät, so gelten die Eigenschaften der letzten vom Scanner erfaßten Scheibe.

Aufbau hinter der Signalmarke:

```
        2 Bytes       Start-LDN dieser Scheibe
        2 Bytes       aa,bb  wie bei SD, für Start-LDN.
        2 Bytes       aa,bb      ''          Start-LDN+1.
        ......
        $00,$00       Stop-marke dieser Scheibe.
```

aa:	$80	Bitfunktion	Station ist rückspulbar (REWIND).
	$40	Bitfunktion	Station kennt OPEN/CLOSE.
	$20	Bitfunktion	Nach dem CR erwartet Station LF. (CR=Carriagereturn, LF=Linefeed)
	$10	Bitfunktion	Station ist dialogfähiges Terminal
	$08	Bitfunktion	Station möchte kein Echo (RS232).
	$04	Bitfunktion	Station kennt 'RM' bzw. 'ERASE'.
	$02	Bitfunktion	Station ist für Ausgabe geeignet.
	$01	Bitfunktion	Station ist für Eingabe geeignet.
bb:	$80	Bitfunktion	Station reagiert auf 'DIR'+'FILES'
	$40	Bitfunktion	Station kennt Formatierbefehl.
	$20	Bitfunktion	Station kennt 'CF' (Change)-Befehl
	$10	Bitfunktion	Hierarchische Verwaltung, 'MKDIR'
	$08	Bitfunktion	SEEK, SYNC, SAVEPOS sind erlaubt.
	$04	Bitfunktion	Reserve
	$02	Bitfunktion	Terminal (RS232c) Editor-parameter
	$01	Bitfunktion	'' ''

```
Beispiel:     DC    $AEB1,$BF95,(10*2+1)*37     Signalmarke
              DC    11                          Start-LDN
              DC.B  $2B,0                       LDN 11
              DC.B  $33,0                       LDN 12
              DC.B  $C7,$E0                     LDN 13
              DC.B  0,0                         Stop-marke
```

Die Station LDN=11 erwartet ein LF nach jedem CR, arbeitet in beiden Richtungen und möchte kein Echo der eingegebenen Daten. Offensichtlich eine typische Host-schnittstelle.

Die Station LDN=12 ist offenbar ein Terminal zum Editieren.

Die Station LDN=13 könnte eine Floppy oder Wechselplatte sein.

---> Die Scheibe 10 entspricht einem automatischen 'SD'-Befehl.

Signalmarke: $AEB1,$BF95,(11*2+1)*37

Das Bedieninterface wird für alle Nutzer um einen oder mehrere Befehle erweitert. Dazu wird mit dieser Scheibe je ein 'Mnemo' der neuen Anweisung sowie die relativierte Sprungadresse definiert.

---> Die definierte Aktion wird auf der Ebene des Bedieninterface mit sehr hoher Priorität ausgeführt. Es ist darum bei zeitaufwendigen Kommandos unbedingt zu prüfen, ob nicht ein Sohnprozess generiert werden sollte.

Aufbau hinter der Signalmarke:

```
DC.B     'String des Befehles'  (ohne Blanks im String!)
(DC.B    0)   Nur wenn String ungerade Anz. Bytes besitzt!
DC.L     Adr-$    Lang relative Ansprungadresse.
.............................................................
DC.B     'String nächster Befehl'
(DC.B    0)   Nur um String auf gerade Bytezahl zu füllen.
DC.L     Adr2-$   Lang relativierte Ansprungadresse.
.............................................................
$FFFF    Stop-marke, Ende dieser Scheibe.
```

Beispiel: Neues Kommando zum Inkrementieren der Zelle $FFA000.

```
        DC        $AEB1,$BF95,(11*2+1)*37    Signalmarke
        DC.B      'INK',$0       Befehlsname, mit $0 aufgefüllt.
        DC.L      INKSR-$        Ansprung
        DC        $FFFF          Stop-marke
        ....      .....
INKSR   ADDQ.B    =1,$FFA000     Eigentliche Operation
        RTS                      Rücksprung Bedieninterface.
```

Nach Anschlag von Ctrl A oder über die XC-Station könnte man nach Einfügen dieser Scheibe in den abgetasteten Bereich den Befehl 'INK' oder 'ink' eingeben. Die 'shell' von RTOS-UH wäre also entsprechend erweitert. Das System nutzt selbst teilweise diese Erweiterungsmöglichkeit, z.B. sind die floppyspezifischen Kommandos durch eine kleine 11-er Scheibe innerhalb des Filehandlers eingebaut.

Das Bedieninterface stellt für die Shellerweiterung eine Reihe von Hilfsfunktionen zur Verfügung. So gibt es die Möglichkeit, den weiteren Text hinter dem Kommando zu analysieren, den Fehlerausgang anzuspringen oder etwa Sohnprozesse zu generieren und mit Parametern (SI=xx etc.) zu versorgen.

---> Die Register D0-D7,A3,A5,A6 können frei benutzt werden.

A0 zeigt beim Einsprung auf die Unterprogrammtabelle des Bedieninterfaces, für z.B. JSR 2(A0) etc.

Signalmarke: $AEB1,$BF95,(11*2+1)*37 (Forts.)

---> A0 zeigt auf die Unterprogrammsprungtabelle des systemseitigen Bedieninterpreters, kann bei Bedarf von $8B8 neu geladen werden, falls es verändert wurde.

---> A1 zeigt auf das Communication-element des Bedieninterfaces, kann notfalls aus dem Task-WSP (über A4) nachgeladen werden mit MOVEA.L $8C(A4),A1.

---> A2 zeigt auf das nächste Zeichen des Eingabepuffers. Damit sind die Traps CSA und QSA zur Textanalyse direkt anwendbar. Muß geordnet zurückgegeben werden!

---> A4 zeigt auf den Task-Workspace des Bedieninterfaces.

---> A5 zeigt auf den Textpuffer eines fertigen Ausgabe-CE des Bedieninterfaces. Mit (A5)+ kann man hier Text ablegen (mit CR abschließen) und mit JSR 28(A0) ausgeben. Die Zielstation ist durch das 'O'-Kommando steuerbar. Nach der Ausgabe wird A5 wieder reinitialisiert.

---> A7 zeigt auf den Stack der Bedientask. Dieser liegt innerhalb des Task-workspace und enthält die Rücksprungadresse in den Bedieninterpreter. Platz für max. 5 Longwords.

---> Alle anderen Register stehen frei zur Verfügung, werden jedoch von den u.a. Unterprogrammen teilweise zerstört.

---> Der Rücksprung in den systemseitigen Interpreter muß mit RTS erfogen, es sei denn, daß einer der über A0 erreichbaren mit JMP anzuspringenden Ausgänge benutzt wird.

Tabelle der über A0 erreichbaren Funktionen (JSR,JMP)

```
JSR  0(A0)  Hier nicht sinnvoll: Aufruf des Interpreters.
JSR  4(A0)  I/O-command processor (Dir,files,mkdir etc.)
            Hinter dem JSR 4(A0) folgende 4 Bytes sind Parameter:
            1. Wort:  Die notwendigen Facilitybits ('SD','DD')
            2. Wort:  links MODE rechts MODE+1 des Comm.el. 'CE'.
JSR  8(A0)  Subtask (son process)- generation.
JSR 12(A0)  Device/File decoder by (A2) to (A3)+
            Ablage füllt ein CE exakt, wenn mit A3 auf STATIO(..)
            zeigend begonnen wird. Work. dir. etc. wird berücks.
JMP 16(A0)  Syntaxfehlerausgang  ('WRONG COMMAND').
--- ------  Nicht innerhalb der Shell-task.
--- ------                  ''
JSR 28(A0)  Make output of Text stored by (A5)+ before.
JSR 32(A0)  Scan on class of character by (A2).
JSR 36(A0)  Write address in A3 as 6 hex. digits to (A5)+
```

Signalmarke: $AEB1,$BF95,(12*2+1)*37

Start und Ende einer beliebigen Zahl von RAM-Sektionen, die unter Verwaltung von RTOS-UH stehen sollen, werden mit dieser Scheibe definiert. Wenn der Abtaster mehrere Scheiben des Typs 12 erfaßt, so werden nur die Parameter der letzten abgetasteten Scheibe gültig. Man muß also in jedem Fall mit dieser Scheibe eine vollständige Definition angeben.

Aufbau hinter der Signalmarke:

```
DC.L    RAM1begin,RAM1end      Speichersektion 1
DC.L    RAM2begin,RAM2end      Speichersektion 2
...                            ....
DC.L    RAMxbegin,RAMxend      Speichersektion x
DC.L    0                      Stopmarke dieser Scheibe
```

---> Die Adressen müssen gerade sein und oberhalb der 2. Adresse müssen noch $20 Zellen RAM sein. (Ende ...E0)

---> Der Adressbereich der Sektion 2 muß überlappungsfrei höher liegen als der von Sektion 1, der von Sektion 3 entsprechend höher sein als der von Sektion 2 etc.

---> Die zwischenliegenden Lücken werden vom Nukleus als scheinbare (unlöschbare) Module namens #NORAM angelegt und erscheinen entsprechend beim S-Befehl.

---> Die letzte Adresse vor der Stopmarke darf zusätzlich noch mit gesetztem LS-Bit versehen werden. Dann wird von der angegebenen Adresse aus bis zum Bus-error oder bis zum ROM oder bis zum Erreichen des Nukleus in 1 kB großen Schritten getestet und die so ermittelte Obergrenze eingesetzt. Achtung: Ende mit $...E1 setzen!

Beispiele:

```
DC      $AEB1,$BF95,(12*2+1)*37     12-er Signalmarke
DC.L    $12000,$0001FFE1            'open end'-spezif.
DC.L    0
```

Das verwaltete RAM erstreckt sich von $12000 bis zum durch Probieren (Bus-error, unveränderlich oder Nukleus erreicht) ermittelten oberen Grenzwert.

```
DC      $AEB1,$BF95,(12*2+1)*37     12-er Signalmarke
DC.L    $800,$3FFE0                 Fixierter Bereich
DC.L    $00080000,$000FFFE0         z.B. VME-System.
DC.L    0
```

Die Grenze von $800 wird automatisch um die Anzahl Zellen, die RTOS-UH dort unten anlegt (ohne evtl. RTOS-UH-Code!), nach oben korrigiert. VORSICHT, wenn RTOS-UH-Code unten im RAM liegt!

Signalmarke: $AEB1,$BF95,(13*2+1)*37

Bei ROM-residenten miteinander kommunizierenden Tasks tritt das Problem auf, einen gemeinsamen Variablenblock zu definieren und diesen beim Systemstart zu initialisieren. Da die Tasks wegen der großen Distanz auf diese Objekte mit absolut langer Adressierung zugreifen müssen, muß die Adresse dieser Modulvariablen fest definiert werden können.

Aufbau hinter der Signalmarke:

```
DC.L   Adr.1              Anfangsadresse des Moduls, gerade.
DC.L   Adr.2   Adresse der (freien) Folgesektion, gerade.
DC.B   'Name6b'                  6 Byte langer Modulname.
DC.L   Data1adresse.  (Adr. erstes von Null versch. Wort).
DC.L   Blocklänge.                (Anzahl der Datenworte).
DC.W   Erstes Datenwort.
....                          ... Datenblock No. 1
DC.W   Letztes Datenwort.
DC.L   Data2adresse.          (Anfang nächster Datenblock).
DC.L   Blocklänge. (Anzahl der Datenworte).
DC.W   Erstes Datenwort.
....                          ... Datenblock No. 2
DC.W   Letztes Datenwort
....
                              ... Datenblock No. x
....
DC.L   0     Statt  Datenadresse:  Stopmarke  der Scheibe.
```

Wichtiger Hinweis: Der angegebene feste Adressraum muß beim Hochlaufen wirklich im verwalteten RAM verfügbar sein, sonst läuft das System nicht an. Alle Worte für die keine Initialdaten angegeben sind, werden vom Nukleus zu Null gesetzt.

Beispiel:

```
DC     $AEB1,$BF95,(13*2+1)*37     Signalmarke.
DC.L   $3000                       Anfangsadresse.
DC.L   $4000                       Nächste FREE- Sektion.
DC.B   'Testmd'                    Name des Modules.
DC.L   $3100                       Datenblock-adr.
DC.L   4                           4 Worte zu initialis.
DC     1            Datum
DC     2             ''
DC     3             ''
DC     4             ''
DC.L   0                           Stopmarke der Scheibe.
```

PEARL-Module für E-PROM-Programme werden erst geladen, dann mit dem Prommer-programm in obenstehende Scheibendaten verwandelt.

Signalmarke: $AEB1,$BF95,(14*2+1)*37

Für nutzerdefinierte Interruptprozesse muß das System eine Möglichkeit zum Besetzen der entsprechenden Adressvektoren schon während der Hochlaufphase bereitstellen.

Aufbau hinter der Signalmarke:

```
DC.W    Adr.1          Vektor- oder Pseudovektoradresse.
DC.W    Jumpadr1-$     Zieladresse für Ansprung relativ.
DC.W    Adr.2          Nächste Vektoradresse o.ä.
DC.W    Jumpadr2-$     Relativierte Zieladresse.
....
DC.W    0              Statt Vektoradresse: Stopmarke.
```

Beispiel: Der Userinterrupt $200 soll nach dem Aufsetzen auf die Adresse XYZ zeigen.

```
        DC    $AEB1,$BF95,(14*2+1)*37   Signalmarke
        DC    $200,XYZ-$-2              -2 wegen 2 Worte im DC!
        DC    0                         Stopmarke
        .....

        DC    XYZMAL-XYZ          Link für den Rückfallmechanism
XYZ     MOVE    IID,-(A7)         Save old  Interrupt-Id  ($7FE)
        MOVE    =$200,IID         Aktuellen  Vectorlink eintrag.
        MOVEM.L  D0-D3,-(A7)
        .....                     Interrupt-code nach  Anwendung
EXIT    MOVEM.L  (A7)+,D0-D3
        MOVE    (A7)+,IID         Altes Vectorlink  wiederholen.
        ORI     =$700,SR          Alle Interrupts stoppen.
        TST     $800  (DPC)       Vorgeschriebener  Interr. exit
        BNE.S   EXIT2             IF DPC == 0 THEN  RTE erlaubt.
        RTE                                  ''
EXIT2   JMP      $908 (DISEX)     DISEX vorgeschrieben statt RTE
        .....
* Rückfall nach hier, wenn Bus-/Adr-/Opcode-exc. im Interrupt:
XYZMAL MOVE ...                   z.B. Coupler abwürgen.
        BRA       EXIT
```

Interruptprogramme müssen den vorgeschriebenen Ausgang benutzen, da sonst der Taskumschalter (Dispatcher) ausfallen kann.

Es können prinzipiell alle Zellen im Bereich bis $7FFE auf diese Weise mit Adresszeigern (4 byte Adr) automatisch versorgt werden, leider kann man damit dann auch u.U. systemeigene Anschlüsse zerstören. Es wird bei Scheiben mit gleichen Vektoradressen immer die letzte vom Abtaster erfaßte Scheibe als gültig genommen.

Man beachte bei dem relativierten Wort, daß das $-Symbol im Assembler immer den Anfang des DC-Blockes bezeichnet, das relativierte Wort aber auf seine eigene Adresse bezogen wird. (Daher die Korrektur mit $-2 im obigen Beispiel).

Signalmarke: $AEB1,$BF95,(15*2+1)*37

Bei der Verwendung von Peripherie-Kopplern kommt es oft zur Notwendigkeit, deren Controlregistern beim Systemstart eine gewisse Anfangsinformation mitzugeben. Auch z.B. das Auslesen einer batteriegepufferten Uhr (um damit die Planungsuhr des Systemes zu stellen) sollte bei jedem Warm- oder Kaltstart erfolgen. Um dies dem Systemprogrammierer zu ermöglichen, sucht der Nukleus nach Abschluß aller sonstigen Aufsetzoperationen mit dem Scanner (Abtaster) nach 15-er Scheiben, um deren Code nacheinander zu exekutieren.

Aufbau hinter der Signalmarke:

```
Maschinencode  fast beliebig, aber kein Tasking, I/O etc.
...            Alle Register bis auf A7 frei verfügbar.
RTS            Rückkehr in den Nukleus.
```

Wichtiger Hinweis: Das System befindet sich noch in einem sogenannten 'Kernelmode' auf der Supervisorebene und kann daher seinen Taskumschalter noch nicht benutzen. Queued I/O etc. ist also keinesfalls möglich. Tritt in dem Scheibencode eine exception (Buserror etc.) auf, so läuft das System unter Umständen mit einem Notstart an. Die im Abtastbereich folgenden 15-er Scheiben werden dann nicht mehr exekutiert, ohne daß dies erkennbar sein muß.

Beispiel: Schreibe das Byte $3C in den Coupler $FF60A2 ein.

```
DC      $AEB1,$BF95,(15*2+1)*37     Signalmarke
MOVE.B  =$3C,$FF60A2                Initialisierung
RTS                                 Zurück (Nukleus)
```

Die Bearbeitung der 15-er Scheiben erfolgt in der Reihenfolge, in der der Abtaster sie findet. In jedem Fall ist die Bearbeitung der letzten 15-er Scheibe auch die letzte Aufsetzoperation des Nukleus. Danach startet unmittelbar der Taskumschalter mit der höchstprioritierten Autostarttask. (Meistens #ERROR)

Für Operationen, die nur beim Kaltstart ausgeführt werden sollen, ist eine eigene Scheibe (18) vorgesehen.

Signalmarke: $AEB1,$BF95,(16*2+1)*37

Die Scheibe gestattet die Ausgabe von ASCII-Text, der in der RTOS-UH Kopfzeile erscheint, sofern das System die standardmäßige ERROR-Task besitzt. Diese Scheibe wird nämlich nicht vom Nukleus bearbeitet, sondern von #ERROR. Die Textzeichen werden nacheinander ausgegeben, ohne daß automatisch Carriage-returns etc. eingefügt werden. Solche Control-zeichen müssen also im Text enthalten sein. Der Text erscheint in der Reihenfolge, in der der Scanner die 16-er Scheiben erfaßt.

Aufbau hinter der Signalmarke:

```
DC.B    'ASCII-Text',$FF        $FF ist Endekennung.
```

Beispiel:

```
DC      $AEB1,$BF95,(16*2+1)*37     Signalmarke
DC.B    $0A,$0D,'Werner Müller's RTOS',$FF
```

Signalmarke: $AEB1,$BF95,(17*2+1)*37

Es ist mit dieser Scheibe möglich, im EPROM oder RTOS-Bootbereich eine globale Marke zu setzen. Der Lader von RTOS-UH kann damit automatisch z.B. EPROM-residente Unterprogramme an das Nutzerprogramm anbinden.

Wann immer der Lader eine nach dem Bearbeiten der Liste noch unbefriedigte Referenz findet, sucht er mit Hilfe des Abtasters den RAM/Bootbereich nach dieser Scheibe ab. Die Scheibe hat also für den Nukleus selbst keine Bedeutung, da sie nur vom Lader bei Bedarf gesucht wird.

Aufbau hinter der Signalmarke:

```
DC.B   'Name6b'        6 Bytes langer globaler Name .
DC     Adr-$           Relativierte Adresse des Symbols.
```

Jede Scheibe dieser Art definiert damit genau ein globales Symbol. Existieren mehrere Scheiben mit dem gleichen Symbol, so verwendet der Lader die erste vom Abtaster erfaßte Scheibe.

Bsp.

```
        DC       $AEB1,$BF95,(17*2+1)*37     Signalmarke 17.
        DC.B     'RANF  '                    Globaler  Name.
        DC       ENTRY-$            relative Einsprungadresse.
        .......                              Beliebige Daten.
ENTRY   ENTR     WSPSZ              zum Beispiel: PEARL-Unter-
        VARF     .....                  Programmkopf (s.E-IV)
        .....
```

Das Label RANF ist jetzt dem Lader so verfügbar, als stünde es in einem der Input-files des Laders.

Auch die Einbaufunktionen des PEARL-Compilers werden mit Hilfe dieser Scheiben angeschlossen, allerdings im Gegensatz zu den normalen Funktionen mit gegen die offene PEARL-Welt geschützten Namen. (z.B. #SSIN, #SSQRT)

Signalmarke: $AEB1,$BF95,(18*2+1)*37

Bei der Verwendung von RAM-Modulen (etwa VME-RAM) kommt es oft zur Notwendigkeit, deren Paritätsbits richtig vorzubesetzen, um einen Bus-error beim späteren Schreib/Lesezugriff zu vermeiden. Auch kann eine 'private' Notiz, daß es sich um einen Kaltstart handelt, nützlich sein. Seltener dagegen wird man Peripherie-Bausteine schon an dieser Stelle initialisieren wollen. Um dieses dem Systemprogrammierer zu ermöglichen, sucht der Nukleus vor Beginn aller sonstigen Aufsetzoperationen mit seinem Scanner (Abtaster) nach 18-er Scheiben, um deren Code nacheinander zu exekutieren.

Aufbau hinter der Signalmarke:

```
Maschinencode  fast beliebig, aber kein Tasking, I/O etc.
...            Alle Register bis auf A7 frei verfügbar.
RTS            Rückkehr in den Nukleus.
```

Wichtiger Hinweis: Das System befindet sich noch in einem sogenannten 'Kernelmode' auf der Supervisorebene und kann daher seinen Taskumschalter noch nicht benutzen. Queued I/O etc. ist also keinesfalls möglich. Tritt in dem Scheibencode eine exception (Buserror etc.) auf, so läuft das System nicht an.

Beispiel: Lösche das RAM von $80000 bis $FFFFFF.

```
        DC      $AEB1,$BF95,(18*2+1)*37     Signalmarke
        LEA     $80000,A1                   Start-adresse
        LEA     $100000,A2                  End-adr.+1
        CLR     D0                  Null vorbereiten, denn CLR
LOOP    MOVE.B  D0,(A1)+            beginnt mit  Read-cycle !!!!
        CMPA.L  A2,A1               Bedingung
        BLT.S   LOOP                Schleifenrücksprung
        RTS                         Rückkehr in den Nukleus.
```

Die Bearbeitung der 18-er Scheiben erfolgt in der Reihenfolge, in der der Abtaster sie findet. In jedem Fall ist die Bearbeitung der 18-er Scheiben allen anderen Operationen vorgelagert.

Für Operationen, die nur beim Warmstart ausgeführt werden sollen, ist eine eigene Scheibe (15) vorgesehen.

This page intentionally left blank.

Belegung der Tastatur unter RTOS-UH

Der nachfolgenden Tabelle sind die Sonderbelegungen der Atari-Tastatur zu entnehmen.

Taste:	Funktion:
Undo	Break-Taste
Shift+Undo	Abort-Taste
Help	Anzeige des Hauptbildschirms
Shift+Help	Anzeige des zuletzt aktiven Bildschirms
Cntr+Cursordown	Wechsel der Schriftfarbe (Nur bei Farbbildschirm)

Im Editor:

Insert	Einfügen eines Buchstaben
Shift+Insert	Einfügen einer Zeile
Delete	Löschen eines Buchstaben
Shift+Delete	Löschen einer Zeile

Terminalemulation des Atari

Die Terminalemulation der Atari-ST-Serie versteht die folgenden Steuerzeichen:

$07	Ring the bell
$08	Cursor einen Schritt nach links (Backspace)
$0A	Cursor einen Schritt nach unten (Linefeed)
$0B	Cursor einen Schritt nach oben (Vertical tab)
$0C	Cursor einen Schritt nach rechts (Formfeed)
$0D	Cursor an den Anfang der Zeile (Carriage return)
$1A	Bildschirm löschen
$1E	Cursor in die linke obere Bildschirmecke setzen (Home)

Die folgenden Escape-Sequenzen sind z.B zum Televideo 910 kompatibel:

Esc *	löscht den gesamten Bildschirm
Esc Y	löscht ab Cursor-Position den Rest des Bildschirms
Esc y	wie Esc Y
Esc T	löscht ab Cursor-Position den Rest der Zeile
Esc t	wie Esc T
Esc =r,c	Cursor positionieren: r=Row c=Column. Zu den Werten für r und c sind immer $20 zu addieren.
Esc 8	Smooth-Scroll einschalten
Esc 9	Auf normalen Scroll umschalten
Esc b	den gesamten Bildschirm invertieren
Esc d	normaler Bildschirm
Esc G 0	Video-Attribut: normale Schrift
Esc G 4	Video-Attribut: inverse Schrift
Esc G 8	Video-Attribut: unterstrichene Schrift
Esc G <	Video-Attribut: inverse und unterstrichene Schrift
Esc E	deutschen Zeichensatz laden
Esc R	US-ASCII Zeichensatz laden

Der Zeichensatz liegt im RAM-Bereich des Rechners. Vom Nutzer kann also auch ein eigener Zeichensatz geladen werden. Der Zeichensatz beginnt mit mit dem Leerzeichen ($20). Jeder Buchstabe ist mit 16 (bei Farbe 8) Byte abgelegt. Den Anfang der Zeichensatztabelle findet man, indem man $00000320 zum Inhalt der Speicherzelle $81E addiert. Der Platz reicht für einen kompletten Zeichensatz mit 255 Zeichen, der ab der angegebenen Stelle ins RAM kopiert werden kann. Nach einem Abort ist der US-ASCII Zeichensatz geladen.

Cursor On/Off

CON / COFF

Syntax: CON oder COFF

Beschreibung: Mit dem Befehl COFF kann der blinkende Cursor des Hauptbildschirms ausgeschaltet werden. Der Befehl CON dient zum Wiedereinschalten des Cursors. Diese beiden Befehle sind bei graphischen Ausgaben auf den Hauptbildschirm sinnvoll einzusetzen. Es ist somit möglich, z.B. eine Hardcopy ohne den störenden Cursor zu machen. Man sollte nicht vergessen, den Cursor wiedereinzuschalten, da sonst z.B. eine Orientierung im Editor schwer fallen dürfte.

Beispiel 1: Ausschalten des Cursors.

Kommando: COFF

Beispiel 2: Einschalten des Cursors.

Kommando: CON

SYNTAX: F1 text ... F10 text

Beschreibung: Mit den Befehlen F1 bis F10 können die Funktionstasten des Atari 520ST programmiert werden. Die Belegung der Tasten bleibt auch nach einem Abort erhalten. Die Tastenbelegung kann jederzeit durch erneuten Aufruf von F1 bis F10 geändert werden.

text: Der Text darf aus ASCII-Zeichen bestehen. Seine Länge ist pro Taste auf 80 Zeichen beschränkt. Um <Control> Zeichen einzugeben, ist das Zeichen '^' zu benutzen.

Beispiel 1: Die Taste F1 soll so belegt werden, daß man die Systemübersicht erhält.

Kommando: F1 ^AS^M

Es wird zuerst <Control A> eingegeben. Dann folgt das S für die Systemübersicht. Als letztes wird ein <CR>, entspricht <Control M>, eingegeben.

Beispiel 2: Es soll bei Betätigung der Taste F9 eine Datei vom Laufwerk F0. in ein ED-File kopiert werden.

Kommando: F9 ^ACOPY F0:XYZ>ED:XYZ

Zuerst wird wieder <Control A> eingegeben, um das Bedieninterface aufzuwecken. Dann folgt der COPY-Befehl. Hier wurde kein <CR> angehängt, so daß der Befehl noch von Hand losgeschickt werden muß.

Hardcopy

SYNTAX: HCPY modulname

Beschreibung: Mit dem Befehl HCPY kann eine Hardcopy auf einen am Centronics-Port angeschlossenen Drucker ausgegeben werden. Die Steuersequenzen passen für IBM-Kompatible Drucker. Soll der Haupt-Bildschirm gedruckt werden, so ist für modulename '#MAIN ' einzusetzen.

modulname: Der Modulname darf aus bis zu 6 Zeichen bestehen. Vor den Namen muß ein '#' gesetzt werden. Das System prüft, ob ein entsprechendes Modul vorhanden ist.

Beispiel 1: Es soll der Haupt-Bildschirm gedruckt werden.

Kommando: HCPY #MAIN

Beispiel 2: Es soll ein nicht sichtbarer Bildschirm, der im Modul '#MAL' liegt, gedruckt werden.

Kommando: HCPY #MAL

Sollte der Drucker nicht IBM oder EPSON kompatibel sein, besteht die Möglichkeit, eine eigene Hardcopy-Routine einzubinden. Diese muß aus eine Modulkopf mit dem Modulnamen 'Hardcp' bestehen und nach dem Modulkopf mit einem NOP beginnen, um exekutiert zu werden. Das Ende der Routine muß ein RTS sein. Im Register A0 erhält man die Anfangsadresse des Bildschirms. Die Register A0-A3 und D0-D7 stehen zur freien Verfügung. Der Stack liegt am Ende des Taskworkspaces und ist mit maximal 7 BSR-Leveln belastbar.

Beispiel:

```
DC.L    0,0     RTOS LINKS      *
DC      $10     TYPE: MODULE    *
DC.B    'Hardcp'  NAME          *
NOP             FOR TEST        *
.                               *
own code of user                *
.                               *
.                               *
RTS             EXIT            *
```

Set Baudrate

SYNTAX: SB baudrate datenbits stopbits parity

Beschreibung: Mit dem Befehl SB kann die Baudrate und das Übertragungsformat der RS 232-Schnittstelle eingestellt werden. Damit ist es möglich, neben einem zweitem Terminal auch Geräte, die nicht mit der Standardrate von 9600 Baud arbeiten, anzuschliessen. Die folgenden Parameter sind zulässig, sie müssen durch Blanks oder Komma getrennt werden:

baurate	19200 Baud	
	9600 Baud	(Standardeinstellung)
	4800 Baud	
	2400 Baud	
	1200 Baud	
	300 Baud	
datenbits:	8 Bit	(Standardeinstellung)
	7 Bit	
	6 Bit	
	5 Bit	
stopbits	2 Bit	(Standardeinstellung)
	1 Bit	
parity	N no parity	(Standardeinstellung)
	O odd parity	
	E even parity	

Beispiel 1: Die Baudrate soll für den Anschluß eines Modems auf 300 Baud, 7 Bits und 1 Stopbit eingestellt werden:

Kommando: SB 300 7 1

Beispiel 2: Es soll wieder die Standardeinstellung gelten:

Kommando: SB 9600 8 2 N

Die Reihenfolge der Parameter muß eingehalten werden, aber restliche Parameter können weggelassen werden. Nach einem Abort gilt wieder die Standardeinstellung.

Systemadressen

videopointer (long) $81A
der videopointer zeigt auf die Startadresse des Hauptbildschirms. Er darf nicht verändert werden.

hide_screen (long) $83A
hide_screen zeigt auf den Bildschirm, der zuletzt genutzt wurde, der Bildschirm muß nicht sichtbar sein.

change_screen (long) $84A
Ist der Wert ungleich Null, wird dieser Wert beim nächsten vertikalen Bildrücklauf in den Videoshifter geschrieben. Das hintere Byte wird dabei nicht berücksichtigt und change_screen wird auf Null gesetzt.

key_answer (8 words) $8E6
Hier können bis zu 8 Worte stehen, die vom Tastaturprozessor als Statusmeldung geschickt werden. Alle Meldungen, auch kürzere, beginnen bei $8E6. Die Meldung wird in der Reihenfolge abgelegt, wie sie von Atari dokumentiert ist. Bei Mausbewegungen z.B. liegt der Mausstatus bei $8E6, die X-Koordinate bei $8E7 und die Y-Koordinate bei $8E8.

Prozeßinterrupts

Drei Prozeßinterrupts sind angeschlossen.

EV 80000000 Dieser Interrupt wird ausgelöst, wenn eine komplette Statusmeldung des Tastaturprozessors vorliegt. Damit kann z.B. auf Bewegungen der Maus von PEARL aus reagiert werden.

EV 40000000 Dieser Interrupt wird bei jedem vertikalen Bildrücklauf gefeuert. Er kann benutzt werden, um synchron zum Bildrücklauf - in der Austastlücke - zwischen verschiedenen Bildschirmen umzuschalten.

EV 20000000 Dieser Interrupt kann extern ausgelöst werden. Er ist an PIN 22 der RS 232 Schnittstelle (ring indicator) angeschlossen und wird mit einer fallenden Flanke ausgelöst. Er kann z.B. mit zusätzlichen I/O Ports benutzt werden, um auf ein externes Ereignis reagieren zu können.

Erweiterte Grafik-Einbaufunktionen für den Atari

Die erweiterten Einbaufunktionen sind nicht im Compiler angeschlossen, müssen daher vor Benutzung im Programm spezifiziert werden. Je nach angeschlossenem Monitor wird eine Auflösung von 640*400 Pixel (Atari SM 124) oder von 640*200 Pixel (4 Farben) unterstützt. Es stehen folgende Funktionen zur Verfügung:

```
CALL SCLOAD(modname,drive,filename);.Bildschirm von Floppy laden.
CALL SCSAVE(modname,drive,filename);.Bildschirm auf Floppy laden.
CALL SCRSET(modname);............Bildschirm im System einrichten.
CALL SCKILL(modname);........Bildschirm aus dem System entfernen.
CALL SCRCHG(modname);............angezeigten Bildschirm wechseln.
CALL PLANE(planenr,colour);..................Farbregister setzen.
CALL PEN(colour);............................Schreibfarbe setzen.
CALL WIDTH(xwidth,ywidth);..................Auflösung feststellen.
CALL CLEAR;..................................Bildschirm löschen.
CALL SPRITL(xpos,ypos,xlen,ylen,modname,array);.....Sprite laden.
CALL SPRITS(xpos,ypos,xlen,ylen,modname,array);.Sprite schreiben.
CALL CIRCLE(xpos,ypos,radius,colour);.............Kreis zeichnen.
CALL TEXT(xpos,ypos,colour,text);......Text an Position ausgeben.
```

```
Parameter :     modname     CHAR(5);
                filename    CHAR(24);
                text        CHAR(80);
                array       BIT(16);
                sonst       FIXED(15);
```

SCLOAD: Von der Diskette (Laufwerk F0 für drive=0, Laufwerk F1 für drive=1) wird der mit drivename bezeichnete File in den mit modname bezeichneten Bildschirm geladen. Es wird geprüft, ob der angegebene Bildschirm eingerichtet ist. Wenn nicht, erfolgt eine Fehlermeldung und der Ladevorgang wird abgebrochen.

SCSAVE: Der mit modname bezeichnete Bildschirm wird auf der Diskette (Laufwerk F0 für drive=0, Laufwerk F1 für drive=1) unter dem mit filename angegebenen Namen abgelegt. Existiert der Bildschirm nicht im System, erfolgt eine Fehlermeldung und der File wird nicht angelegt.

SCRSET: Es wird ein verdeckter Bildschirm im System eingerichtet. Damit ist es möglich, Graphiken im Hintergrund zu erstellen. Der Bildschirm erhält den Namen, der mit modname übergeben wird. Vor modname wird automatisch ein '#' eingefügt, um ein Entladen mit UNLOAD zu verhindern. In den Taskworkspace der aufrufenden Task wird die Adresse des Bildschirms eingetragen. Diese Adresse wird von SETPIX, LINE, CIRCLE und CLEAR benutzt. Ist ein Bildschirm mit modname schon im System vorhanden, wird kein zweiter eingerichtet, sondern es erfolgt nur die Eintragung der Adresse. Die Adresse wird bei jeder Aktivierung der Task gelöscht, so daß z.B. eine zyklisch eingeplante Task jedesmal SCRSET aufrufen muß. Steht nicht mehr genügend Speicher zur Einrichtung des Bildschirms zur Verfügung, wird die Task suspendiert. Sie läuft bei genügend freiem Speicher weiter.

SCKILL: Mit SCKILL kann ein verdeckter Bildschirm aus dem System entfernt werden. Dabei wird automatisch die Adresse des Bildschirms im Taskworkspace gelöscht. Diese Routine steht auch als Bedienkommando zur Verfügung. Dann muß vor dem Modulnamen aber das '#' mit eingegeben werden. Vorsicht ist geboten, wenn ein Bildschirm entfernt wird, in dem noch gemalt wird: es kann zu Systemabstürzen kommen!

SCRCHG: Der angegebene Bildschirm wird zur Anzeige gebracht. Wird das Umschalten des Schirmes mit dem Bildrücklauf synchronisiert, können, je nach Anzahl der Bilder, kleine Trickfilme realisiert werden. Dieser Befehl steht auch als Shell-Erweiterung zur Verfügung. Dann muß beim modname aber ebenfalls das '#' mit angegeben werden.

PLANE: Die Farbinformation in den Farbpalettenregistern wird geändert. Damit ändert sich die Farbe des gesamten Bildschirmes für alle Bilder. Bei schwarz/weiß-Darstellung wird nur das letzte Bit im Plane-Register berücksichtigt. In der mittleren Auflösung, die von RTOS-UH unterstützt wird, werden die Plane-Register 0-3 berücksichtigt. Die Organisation der Farbpalettenregister sieht folgendermaßen aus:

```
FEDCBA9876543210
-----rrr-ggg-bbb
```

Die mit r bezeichneten Bits bestimmen den Rot-Anteil, die mit g bezeichneten den Grün- und die mit b bezeichneten den Blau-Anteil der Farben. Je größer der Wert dieser drei Bits ist, um so intensiver ist die Farbe.

PEN: Es wird die Schriftfarbe festgelegt. Die Farbe 0 entspricht dem Hintergrund. Mit den Farben 1-3 kann das entsprechende Paletten-Register angewählt werden.

WIDTH: Die aktuelle Bildschirmauflösung wird in die Variablen xwidth und ywidth eingetragen (z.B. hohe Auflösung xwidth=640, ywidth=400). Damit ist es möglich, Programme zu schreiben, die bei verschiedenen Auflösungen lauffähig sind.

CLEAR: CLEAR löscht den Bildschirm, der im Taskworkspace mit SCRSET eingetragen wurde. Ist kein eigener Bildschirm eingerichtet, so benutzt CLEAR, ebenso wie LINE, SETPIX und CIRCLE, den Hauptbildschirm.

SPRITL: Von dem angegebenen Bildschirm wird ein Ausschnitt der Größe xlen, ylen in ein zweidimensionales BIT(16) Pearl-Feld gerettet. Es wird geprüft, ob der Bereich innerhalb des Bildschirms liegt, wenn nicht, erhält man die Fehlermeldung 'Out_of_screen'. Desweiteren wird überprüft, ob der Ausschnitt in das angegebene Pearl-Feld paßt. Dabei werden die einzelnen Feldgrenzen überprüft. Reicht eine nicht, erhält man die Fehlermeldung 'X-Dim._of_Array_to_little' oder 'Y-Dim._of _Array_to_little'. Sind xlen/ylen kleiner oder gleich Null, wird die Meldung 'Xlen_negativ' oder 'Ylen_ negativ' ausgegeben. Bei jedem der beschriebenen Fehler wird die aufrufende Task suspendiert und bei einem Continue terminiert.
Die X-Dimension des Feldes muß die Größe (xlen/16) haben. Wenn der Rest dieser Division ungleich Null ist, muß das Feld ein Wort größer sein. Die Y-Dimension hat die Größe von ylen. Es wird nur der tatsächlich angegebene Ausschnitt gerettet, ist z.B. Xlen=5, so werden in X-Richtung 5 Bits linksbündig in das Feld geschrieben.

Beispiel: Bildschirm:

```
oooooxxxxxxooo...
oooooxxxxxxooo...
oooooxxxxxxooo...
................
```

Aufruf: CALL SPRITL(4,0,6,3,'MAIN ',Feld);

Feld :

```
xxxxxx....
xxxxxx....
xxxxxx....
..........
```

DCL Feld (1,5) BIT(16);

In dem Feld liegen die Bits wie auf dem Bildschirm. Um in dem obigen Beispiel z.B. auf das Pixel mit den Koordinaten (3,1) *(hervorgehoben)* zuzugreifen, muß mit

BIT=Feld(1,2).BIT(4); gearbeitet werden.

Bei Farbdarstellung werden beide Planes gerettet. Zuerst wird das erste Plane und danach das Zweite komplett im Feld abgelegt. Es wird also die doppelte Y-Länge des Feldes benötigt. Da die Auflösung in Y-Richtung bei Farbe auch nur 200 Pixel beträgt, kann der gesammte Bildschirm sowohl bei Farbe wie auch bei Schwarz/Weiß in ein Feld der Größe (40,400) BIT(16) ge-rettet werden.

Anmerkung: Mit SPRITL/SPRITS können Bildausschnitte über den Bildschirm bewegt werden. Dazu muß ein Bildausschnitt mit SPRITL gerettet werden, dann mit dem Ausschnitt überschrieben und zuletzt mit SPRITS restauriert wer-den. Wenn sowohl die X-Koordinate als auch Xlen ohne Rest durch 8 teilbar sind, wird der Transfer in Hochge-schwindigkeit durchgeführt! Je größer der Bereich ist, desto mehr wirkt sich der Geschwindigkeitsvorteil aus. Wird z.B. der ganze Bildschirm in ein Feld geschrieben, geht es ca. 6 mal schneller, als wenn ein Bit weniger gerettet würde.

SPRITS: Ein Feld[ausschnitt] wird auf einen Bildschirm geschrieben. Alle Überprüfungen erfolgen genauso, wie bei SPRITL.

CIRCLE: Es werden Vollkreise auf den Bildschirm, der im Taskworkspace eingetragen ist, gezeichnet. Ist kein Bildschirm eingetragen, wird auf den Hauptbildschirm gezeichnet. Ist der Kreis größer als der Bildschirm, werden nur die entsprechenden Segmente gezeichnet. Eine Zerstörung anderer Speicherstellen ist ausgeschlossen.

TEXT: Es kann ein beliebiger Text auf den Bildschirm, der im Taskworkspace eingetragen ist, ausgegeben werden. Der Text darf maximal 80 Zeichen lang sein. Er wird solange ausgegeben, bis nur noch Leerzeichen kommen. Diese werden unterdrückt, um auch kürzere Texte ausgeben zu können. Der Text wird nur auf den Bildschirmspeicher geschrieben, es kann nicht zu einer Systemzerstörung kommen. xpos/ypos bezeichnet das linke obere Pixel der Zeichenmatrix des ersten Zeichens.

Anmerkung: Ist modname='MAIN ', so wirken die Routinen auf den Hauptbildschirm. Der Aufruf von SCRSET ist nicht nötig. Wird bei den Routinen LINE, SETPIX, CIRCLE und TEXT für die Farbe eine negative Zahl eingesetzt, werden die Bits auf dem Bildschirm invertiert. Durch zweimaligen Aufruf mit den gleichen Koordinaten kann z.B. eine Linie wieder gelöscht werden.

Beispiel:

```
PROBLEM;
SPC CLEAR ENTRY GLOBAL;
SPC WIDTH ENTRY (FIXED(15),FIXED(15)) GLOBAL;
SPC SCRSET ENTRY (CHAR(5)) GLOBAL;
 .
xxxx:TASK;
DCL (xwidth,ywidth) FIXED(15);
...
   CALL WIDTH(xwidth,ywidth);
   ...
   CALL SCRSET('BILD ');
   CALL CLEAR;
    .
   CALL LINE(0,0,xwidth,ywidth,1);
    .
   CALL SCRCHG('BILD ');
...
END;
...
MODEND;
```

Die ATARI -Harddisk

Zur erstmaligen Benutzung der ATARI-Hardisk ist die Verwendung der auf der Utility-Diskette mitgelieferten Verwaltungsprogramme WINAS und WINPS
erforderlich.

Um die ATARI-Winchester unter RTOS-UH zu benutzen, ist in folgender Weise zu verfahren:

1) Formatieren und Partitionieren sie die Winchester unter TOS wie im TOS-Manual beschrieben.

2) Starten Sie RTOS-UH und laden Sie das RTOS-UH-Harddisk-Verwaltungsprogramm von der im Laufwerk 0 eingelegten Utility-Diskette mit

   LOAD F0.WINAS+F0.WINPS

   Rufen Sie es mit

   SETUP

   auf. Jetzt werden Ihnen für jede Partition die Länge und die Anzahl der Blöcke angezeigt.

3) Auf die Frage

   EINE PARTITION UMSETZEN:

   antworten Sie mit 1, 2, 3 oder 4, je nach gewünschter Partition-Nummer.

4) Bei der Frage

   PARTTION ID:

   antworten sie mit UHF, wenn Sie eine Partition auf RTOS-UH umschalten möchten. Wenn Sie eine Partition auf TOS zurückschalten wollen, geben Sie hier GEM ein.

5) Schritt 3 und 4 wiederholen Sie solange, bis alle Partitionen die richtige ID haben. Dann antworten Sie auf die Frage

   EINE PARTITION UMSETZEN:

   nur mit der <Return>-Taste.

6) Sind nun alle Partition auf die korrekte ID gesetzt, so antworten Sie auf die Frage

   NEUEN ROOT SEKTOR AUF DIE PLATTE SCHREIBEN(J/N):

   mit J. Der neue Root-Sektor wird nun auf die Winchester übertragen und das Programm verabschiedet sich mit der Meldung

   DAS WAR'S

   Sind die Partitionen noch nicht auf die richtige ID gesetzt, so antworten Sie mit N. Das Programm verabschiedet sich

daraufhin mit der Meldung

DANN EBEN NICHT

und schreibt keinerlei Informationen auf die Harddisk. Sie haben den Initialisierungsvorgang hiermit abgebrochen und müssen ggf. bei 1) neu beginnen.

7) Notieren Sie sich nun die FORMAT-Angaben für die Partitionen auf einem Blatt Papier.
Der RTOS-UH-Harddisk-Driver ordnet der ersten auf UHF gesetzten Partition den Systemname H0: zu, der zweiten auf UHF gesetzten Partition den Systemnamen H1:, etc.

8) W I C H T I G
Führen Sie jetzt einen Warmstart des RTOS-UH-Systems mit

<Shift>+<Undo>

aus.
(Wissen Sie jetzt, warum Sie die FORMAT-Angaben auf dem Papier notieren sollten ?)

9) Jetzt können Sie mit

FORM D H0:formatangabe0

die erste Partition initialisieren.
Dabei gehen alle Information auf dieser Partition verloren !
Wenn Sie weitere Partitionen eingerichtet haben, initialisieren Sie diese mit

FORM D H1:formatangabe1
FORM D H2:formatangabe2
FORM D H3:formatangabe3

Die auf die Laufwerksangabe foldende Information (formatangabe) übernehmen Sie bitte den Angaben des SETUP-Programm's.

FEHLERMÖGLICHKEITEN:

Wenn das SETUP-Programm keine Antwort vom Laufwerk erhält, läuft es nicht an. Es muß dann mit UNLOAD entfernt werden, sonst wird eine Fehlermeldung ausgegeben.

Im Atari wurd die zusätzliche Funktion CURPOS implementiert, die die aktuelle Cursor-Position zurückgibt. Sie muß, wie alle nicht zum PEARL-Standard gehörenden Funktionen im Problemteil folgendermaßen spezifiziert werden:

```
SPC CURPOS ENTRY(CHAR(1) IDENT,CHAR(1) IDENT) GLOBAL;
```

Die Funktion gibt an die beiden Parameter (line,column) die beim Aufruf aktuelle Position des Cursors zurück. Der Zeilen- und Spaltennummer sind dabei mit dem Summand 32 (hexadezimal $20) beaufschlagt, damit die erhaltenen Werte mit der in Kapitel F-I-2 beschriebenen Terminalemulationfunktion 'ESC =',line,column direkt zur Positionierung des Cursors genutzt werden können.

Beispiel:

```
MODULE M; SYSTEM;......
PROBLEM; ..SPC CURPOS ENTRY(CHAR(1) IDENT,CHAR(1) IDENT)GLOBAL;
....
TEST: TASK;
DCL (line,column) CHAR(1);
     ..........
     CALL CURPOS (line,column); /* Merke Cursor-Position*/;
     .....
     PUT ......  /*  ausgaben einer Meldung  an  eine  andere
                     Bildschirmposition */;
     ......
     PUT  TOCHAR(27),'=',line,column  TO  ...  /*  CURSOR  auf
                          gemerkte Position zurücksetzen */
     ......
END;
.....
```

Allgemeines

Die folgende Beschreibung unter dem Stichwort MONITOR bezieht sich auf zwei Programme:

1.) Das Programm MONI, das als normale Task im Echtzeit-Betriebssystem läuft und von der Tool-Diskette geladen wird.

2.) Das Programm MONITOR, das durch den Taskaufruf MONITOR gestartet wird und nicht im Multitasking-Betrieb läuft, sondern alle aktiven Tasks abbricht.

Beim Aufruf werden alle erlaubten Befehle in zwei Menuezeilen angezeigt. Das > signalisiert die Eingabebereitschaft auf der Kommando-Ebene. Alle Funktionen werden durch zwei Zeichen aufgerufen. Diese beiden Buchstaben sind in jedem aufgeführtem Befehl groß geschrieben, können aber in Groß- oder Kleinschrift eingegeben werden. Der Monitor arbeitet dialogorientiert. Sämtliche Ein- und Ausgaben sind hexadezimal. Werden Abfragen nach Start- oder Endadressen nur mit RETURN quittiert, übernimmt das Programm Default-Werte oder die bisherigen Eingaben. Längere Bildschirmausgaben können durch Tastendruck abgebrochen werden. Die ESCAPE-Taste führt prinzipiell auf die nächsthöhere Ebene, verläßt also eine erwartete Eingabe oder bricht das Kommando ab.

Unlogische Eingaben werden neu angefordert, zum Beispiel der Versuch, eine ungerade Adresse als Start für den Disassembler zu wählen.

Der Befehlsvorrat der beiden Versionen ist unterschiedlich. Die Debugging-Kommandos (GO, TraceProg, ShowRegister und so weiter) sind nur unter MONITOR verfügbar, die Protokollierungs-Funktionen (EdProt, PrinterProt) nur unter MONI. Bei der Beschreibung der einzelnen Befehle ist dies jeweils vermerkt, bei Befehlen ohne einen derartigen Vermerk kann man davon ausgehen, daß dieses Kommando in beiden Versionen verfügbar ist.

Beachten Sie bitte, daß es bei fast allen Kommandos möglich ist, "lebenswichtige" Adressen des Monitors oder des Systems zu überschreiben. Ein Absturz ist also nicht vorprogrammiert, aber voll und ganz in der Hand des Benutzers.

BESONDERHEITEN MONITOR

Die Vektoren zur Fehler-Behandlung zeigen alle auf den Monitor. Es wird jeweils die Art des Fehlers gemeldet (Buserror, Adresserror etc.), eine genauere Analyse ist anschließend über das ShowReg-Kommando möglich. Die Zuordnung der ausgegebenen Meldungen zu den jeweiligen Vektoren finden Sie unter dem Stichwort "Error-Meldungen".

Der Monitor initialisiert beim Start zwei Buffer ($400-$7ff und einen 1KByte-Bereich vor dem Video-RAM). Daten in diesen Bereichen gehen also verloren. Der Monitor arbeitet unabhängig vom RTOS und deshalb nicht im Multitasking-Betrieb. Bei seinem Aufruf werden laufende Tasks abgebrochen.

BESONDERHEITEN MONI

MONI wird von der Tool-Diskette mit LOAD Fx.MONISR geladen und mit MONI aktiviert. Die Task MONI schaltet das Bedien-Interface nicht ab. Es ist deswegen über CTRL und A jederzeit erreichbar. Dies mag auf den ersten Blick ungewohnt erscheinen, macht aber beispielsweise folgende, praktisch sehr hilfreiche, Vorgehensweise möglich:

- MONI starten
- CTRL und a
- mit dem S-Kommando des Systems feststellen, wo das zu untersuchende Modul im Speicher steht
- mit ESCAPE die Kommando-Ebene des Monitors aufrufen und den entsprechenden Bereich disassemblieren.

Um die Protokoll-Funktion EdProt (Protokoll auf ein Editor-File) nutzen zu können, sollte das ebenfalls auf der Tool-Diskette vorhandene File EDMONSR geladen und gestartet sein (LOAD Fx.EDMONSR -- EDMON). Näheres dazu finden Sie bei der Beschreibung des EdProt-Kommandos.

Da bei jedem Zugriff auf den Speicher Dispatcher und Interrupts kurz abgeschaltet werden, können auch Adressen gelesen und/oder beschrieben werden, die sonst im User-Mode nicht zugänglich sind. Dadurch werden natürlich sämtliche Funktionen verlangsamt, was aufmerksamen Menschen wahrscheinlich schon bei der Bildschirmausgabe auffällt. Bei Befehlen wie COmpareMem oder HuntBytes sollte deswegen nicht immer pauschal der ganze Speicher durchsucht werden, weil dies zu spürbaren Wartezeiten führen kann. Ein Zugriff auf nicht existierende Adressen erzeugt einen Bus-Error, der die Task MONI suspendiert.

Kurzübersicht Kommandos

CO	COmpMem	Speicher-Bereiche vergleichen
DA	DissAss	Dissassemblieren
DM	DumpMem	Memory-Dump in Hex-/ASCII-Darstellung
FM	FillMem	Speicher-Bereiche füllen
HB	HuntBytes	Suchen von Byte-/ASCII-Kombinationen
MM	MoveMem	Speicher-Bereiche verschieben
SM	SetMem	Speicher-Inhalt verändern
??	Help	Befehlsliste zeigen lassen
Nur MONI:		
CD	ContinousDump	Ständiges Dump einer Adresse
EY	EdprotokollYes	Ausgabeprotokoll in ED.-File einschalten
EN	EdprotokollNo	Ausgabeprotokoll in ED.-File ausschalten
PP	PrinterProtokoll	Ausgabeprotokoll auf Drucker
QM	QuitMoni	Task MONI terminieren
Nur MONITOR:		
GO	GOprog	Programm in Echtzeit starten
RT	RTos	Kaltstart des RTOS-Systems
RW	RtosWarm	Warmstart des RTOS-Systems
SB	SetBrks	Break-Point setzen
SR	ShowRegs	Register-Inhalte ausgeben
SS	SinglStep	Einzelschritt-Modus
TP	TraceProg	Programm im Trace-Mode starten

Befehl:

CD	ContinousDump	Ständiges Dump einer Speicherstelle

Beschreibung:

Der Inhalt einer Adresse wird byteweise ausgegeben. Abbruch mit ESCAPE.

Hinweise:

Der byteorientierte Zugriff erlaubt auch das Auslesen von Adressen der I/O-Bausteine.

Nur MONI!

Befehl:

CO COmpMem Speicher-Bereiche vergleichen

Beschreibung:

Der Block von 'Startadresse' bis 'Endadresse' wird mit dem Block gleicher Länge ab 'Zieladresse' verglichen. Unterschiedliche Inhalte werden durch Ausgabe der korrespondierenden Adressen und der unterschiedlichen Inhalte angezeigt. Die Ausgabe kann durch jede beliebige Taste abgebrochen werden, nach RETURN können neue Adressen eingegeben werden, ESCAPE zeigt Ihnen das Prompt-Zeichen.

Befehl:

DA DissAss Dissassemblieren

Beschreibung:

Jeweils 16 Befehle ab 'Startadresse' werden in mnemonischer Form dargestellt. Immediate-Adressierung wird durch das Doppelkreuz (#) gekennzeichnet. Ungültige Codes werden in der Form .dc.w hexzahl und, falls möglich, in ASCII-Form dargestellt. Die Ausgabe kann durch Tastendruck unterbrochen bzw. fortgesetzt werden, nach RETURN wird eine neue Startadresse angefordert. ESCAPE führt zurück zum Kommando-Modus.

Befehl:

DM	DumpMem	Memory-Dump in Hex-/ASCII-Darstellung

Beschreibung:

Eine Page (255 Bytes) des Speicher-Bereichs ab 'Startadresse' wird als Hex-Dump ausgegeben, rechts daneben in ASCII-Darstellung (soweit möglich). ESCAPE führt zurück zum Kommando-Modus, nach RETURN kann eine neue Startadresse eingegeben werden, bei Betätigen einer beliebigen anderen Taste wird die nächste Page angezeigt.

Befehl:

EN	EdprotokollNo	Funktion 'Alle Ausgaben auf File ED.PROT protokollieren' ausschalten

Beschreibung:

Das EN-Kommando schaltet die Funktion 'Protokollierung aller Ausgaben auf das File ED.PROT' aus.

Hinweise:

Alles weitere siehe unter EY!

Nur MONI!

Befehl:

EY	EdprotokollYes	Funktion 'Alle Ausgaben auf File ED.PROT protokollieren' einschalten

Beschreibung:

Das EY-Kommando schaltet die Funktion 'Protokollierung aller Ausgaben auf das File ED.PROT' ein. Dazu muß vorher das Programm EDMONSR geladen und gestartet sein:
LOAD Fx.EDMONSR
EDMON
Die Task EDMON befindet sich dann, solange keine Daten vom Monitor kommen, im Zustand "I/O?". Der Transfer der Daten läuft über die Datenstation VI/VO unter dem Namen EDP.

Hinweise:

Das Senden der Daten über VO.EDP beinhaltet die Möglichkeit, VI.EDP von einer anderen Task als EDMON bedienen zu lassen, beispielsweise einem COPY-Befehl, der als Ziel ein Floppy-File hat.

Nur MONI!

Befehl:

FM FillMem Speicher-Bereiche füllen

Beschreibung:

Der Bereich von 'Startadresse' bis 'Endadresse' wird mit einem String gefüllt, der, wie immer, maximal 60 Zeichen lang sein kann und aus einer beliebigen Mischung von Hex- und ASCII-Eingaben bestehen darf. Nach RETURN können neue Adressen und/oder Strings eingegeben werden, ESCAPE beendet das Kommando und führt zurück zur Kommando-Ebene.

Beispiel: Startadresse: C0000 Endadresse: C0003
Bytefolge: "Dies ist das Haus vom Nicolaus"

füllt die Adressen von $C0000 bis $C0003 mit "Dies".

Hinweise:

Bei der Eingabe von Hexadezimal-Zahlen muss die Anzahl der Hexdigits gerade sein, es sei denn, die Byte-Folge besteht nur aus einem Hexdigit. Also $0C00, nicht $C00!

Befehl:

GO	GOprog	Programm in Echtzeit starten

Beschreibung:

Ein Programm ab 'Startadresse' wird ausgeführt. Die einzige Kontrollmöglichkeit besteht durch Setzen von Break-Points (siehe unten). Vor dem Start werden Daten-, Adress- und Status-Register aus dem Buffer geladen (siehe SR-Kommando) und eventuell gesetzte Break-Points eingesetzt (siehe SB-Kommando).

Nur MONITOR!

Befehl:

HB HuntBytes Suchen von Byte-/ASCII-Kombinationen

Beschreibung:

Es wird nach 'Startadresse' und 'Endadresse' des zu durchsuchenden Bereichs gefragt. Der durch 'Bytefolge' angeforderte String darf maximal 60 Zeichen lang sein, hexadezimale und ASCII-Eingaben können beliebig gemischt werden. ASCII-Zeichen bzw. Strings müssen in doppelte Hochkommata (") eingeschlossen sein. Gefundene Adressen werden durch 'found in: adress' ausgegeben. Die Ausgabe kann durch Tastendruck abgebrochen werden. ESCAPE führt zurück zum Kommando-Modus, nach RETURN können neue Eingaben gemacht werden.

Beispiel:

Startadresse: 8000 Endadresse: 9000
Bytefolge: 4e41"Mon.Doc"8a

Der Adress-Bereich von $8000 bis $9000 wird nach der Bytefolge 4e414d6f6e2e446f638a durchsucht.

Hinweise:

Bei der Eingabe von Hexadezimal-Zahlen muss die Anzahl der Hexdigits gerade sein, es sei denn, die Byte-Folge besteht nur aus einem Hexdigit. Also $0C00, nicht $C00!

Befehl:

MM MoveMem Speicher-Bereiche verschieben

Beschreibung:

Der Block von 'Startadresse' bis 'Endadresse' wird zur 'Zieladresse' verschoben. Nach Return können neue Adressen eingegeben werden, ESCAPE beendet den Befehl.

Beispiel:

Startadresse: F8000 Endadresse: FA000 Zieladresse: F8001

verschiebt den oberen Teil des Bildschirms der 1MByte-Version um ein Byte.

Befehl:

PP	PrinterProtokoll	Alle Ausgaben auf Drucker protokollieren

Beschreibung:

Das PP-Kommando toggelt ein Flag für die Protokollierung aller Ausgaben auf den Drucker, schaltet also beim ersten Aufruf das Drucker-Protokoll ein, beim nächsten aus und so fort.

Nur MONI!

Befehl:

QM QuitMon Task MONI terminieren

Beschreibung:

Die Task MONI wird in den Zustand "DORM" versetzt.

Hinweise:

Dieser Befehl hat keinen Einfluß auf den Zustand der Task EDMON!

Nur MONI!

Befehl:

RT	RTos	Sprung zum RTOS-System

Beschreibung:

Neustart des RTOS-Systems. Es handelt sich dabei um einen sogenannten Kaltstart, das heißt, das System wird komplett neu initialisiert. Sämtliche User-Daten gehen verloren.

Nur MONITOR!

Befehl:

RW RtosWarm Warmstart des RTOS-Systems

Beschreibung:

Rückkehr ins System in Form eines Warmstarts. Sämtliche Daten bleiben wie vor dem Sprung in den Monitor erhalten.

Nur MONITOR!

Befehl:

SB SetBrks Break-Point setzen

Beschreibung:

Maximal 10 Break-Points können gesetzt werden. Diese werden vor dem Start eines Programms durch GO (siehe oben) eingesetzt, die Befehle an diesen Adressen werden gerettet. Läuft das Programm auf einen dieser Break-Points, werden in der Fehlerbehandlungs-Routine zunächst die geretteten Befehle wieder eingesetzt, und dann wird in den Single-Step-Modus (siehe dort) verzweigt. Ein Break-Point wird gelöscht, indem er auf Adresse 0 gesetzt wird. Nach der Eingabe von SB wird jeder Break-Point einzeln angezeigt. Durch Drücken von B kann dieser verändert werden, jeder andere Taste zeigt den nächsten Break-Point an.

Beispiel:

```
0A0000 B: 0
000000
00E000 B: C000
000000 B: F0000
...
```

Break-Point 0 wurde nach Drücken von B gelöscht, Break-Point 1 übergangen, Break-Point 2 von $E000 auf $C000 verändert, Break-Point 4 neu gesetzt auf $F0000. (Die restlichen sechs Ausgaben sind durch die drei Punkte gekennzeichnet.)

Hinweise:

Will man ein Programm durch Setzen von Break-Points und den GO-Befehl testen, sollte man sicherstellen, daß diese auch erreicht werden. Denn der Monitor hat dann keine weitere Kontrolle über die Programmausführung.

Nur MONITOR!

Befehl:

SM SetMem Speicher-Inhalt verändern

Beschreibung:

Maximal 60 Bytes ab 'Start-Adresse' können mit einer beliebigen Kombination von Hex- und ASCII-Zeichen beschrieben werden. Nach Eingabe der Startadresse werden 16 Bytes in Hex- und ASCII-Darstellung angezeigt, darunter wird die Eingabe erwartet. RETURN beendet die Eingabe, ESCAPE wirkt in dieser Zeile wie UNDO. Der neue Inhalt wird zur Kontrolle noch einmal angezeigt. Durch RETURN kann eine neue Startadresse gewählt werden, durch ESCAPE gehts zurück zum Kommando-Modus, nach Betätigen jeder anderen Taste können die nächsten 16 Bytes editiert werden.

Hinweise:

Bei der Eingabe von Hexadezimal-Zahlen muss die Anzahl der Hexdigits gerade sein, es sei denn, die Byte-Folge besteht nur aus einem Hexdigit. Also $0C00, nicht $C00! Vorsichtige Menschen sollten nur jeweils 16 Bytes editieren, da die Kontroll-Ausgabe nur diese anzeigt.

Befehl:

SR ShowRegs Register-Inhalte ausgeben

Beschreibung:

Die Daten- und Adress-Register, das Status-Register, der Programm-Zähler, der User-Stackpointer, der Super-Visor-Stackpointer und der durch den Programm-Zähler bezeichnete Befehl werden ausgegeben. Bis auf den Befehl selbst können alle Werte verändert werden. Es gilt folgende Syntax:

An: Adress-Register n (0<=n<8) kann verändert werden.
Dn: Daten-Register n (0<=n<8) kann verändert werden.
PC: Programm-Zähler setzen.
US: User-Stackpointer setzen.
SP: Supervisor-Stackpointer setzen.
SR: Status-Register verändern.

Der Doppelpunkt muß mit eingegeben werden!

Nach jeder Änderung werden die Register wieder angezeigt. Rückkehr zum Kommando-Modus wie immer mit ESCAPE. Es handelt sich bei diesen Registern, dem Programm-Zähler usw. natürlich nicht um die des Monitors. Zunächst wird lediglich ein Buffer mit diesen Werten beschrieben. Diese Werte werden dann gesetzt für den Trace- und Single-Step-Befehl und zu Beginn eines GO-Befehls. Außerdem werden nach jeder Exception die Register-Werte in diesen Buffer kopiert, um den aufgetretenden Fehler analysieren zu können.

Beispiel:

d0:1

setzt Daten-Register 0 auf 00000001

Hinweise:

Vorsicht bei Manipulationen des Supervisor-Stackpointers bzw. von A7 (was auf dasselbe hinausläuft)! Es ist ein gesonderter Stack-Bereich für die zu "tracenden" Programme vorgesehen. So wird auch der Programmzähler und das Status-Register auf diesem Stack abgelegt. Eine unbedachte Änderung des Stackpointers kann also leicht den Stack des Monitors oder andere wichtige Arbeitsadressen zerstören. Dies gilt natürlich auch für Programme, die durch Trace oder SingleStep getestet werden. Außerdem ist der Stack-Bereich für diese Programme auf 400 Bytes begrenzt.

Nur MONITOR!

Befehl:

SS	SinglStep	Einzelschritt-Modus

Beschreibung:

Ab 'Startadresse' wird ein Programm im Einzelschritt-Modus ausgeführt. Nach jedem Befehl werden alle Register, Status, die Stackpointer, der Programmzähler und der nächste Befehl angezeigt. Durch RETURN kann der Programmzähler verändert werden, ESCAPE führt zurück zur Kommando- Ebene, bei Betätigen einer beliebigen anderen Taste wird jeweils der nächste Befehl ausgeführt.

Hinweise:

Das getestete Programm wird real ausgeführt, schreibt also eventuell auch in Adressen des Monitors.

Nur MONITOR!

Befehl:

TP	TraceProg	Programm im Trace-Mode starten

Beschreibung:

Ein Programm wird ab 'Startadresse' im Trace-Modus ausgeführt. Überschreitet der Programmzähler die angegebene 'Endadresse', werden die Register angezeigt, und es wird in den Single-Step-Modus (siehe unten) verzweigt. Vor dem ersten Befehl werden Daten-, Adress- und Status-Register mit den Werten aus dem Buffer geladen (siehe SR-Kommando).

Nur MONITOR!

Befehl:

?? Help Befehlsliste zeigen lassen

Beschreibung:

Die möglichen Befehle mit den entsprechenden Kommandos in Großbuchstaben werden angezeigt. Dasselbe geschieht übrigens auch bei Eingabe ungültiger Kommandos.

'Error'-Meldungen

Nur MONITOR!

Soweit nicht einzeln vermerkt, werden die Vektoren bis inclusive Vektor-Nummer $3f auf eine Routine gelegt, die die Meldung 'unuerror' ausgibt. 'unu' steht dabei für 'unused'.

Meldung	Fehlerart	Vektornummer
buserror	Bus-Fehler	$02
adrerror	Adress-Fehler	$03
ibferror	Illegaler Befehl	$04
dvzerror	Division durch Null	$05
chkerror	CHK-Befehl	$06
trverror	TRAPV-Befehl	$07
prverror	Privilegverletzung	$08
aoperror	Opcode $Axxx	$0A
foperror	Opcode $Fxxx	$0B
niierror	Non-init. Interrupt	$0F
faierror	Falscher Interrupt	$18

Der Trace-Vektor ($09) wird natürlich vom Trace- und Single-Step-Kommando benutzt.

This page intentionally left blank.

```
000000                  *-------------------------------------------------------*
000000                  *       R  I  O  S  -  U  H  -  Atari ST-Serie          *
000000                  *-------------------------------------------------------*
000000                  *       (c)      IEP - Hannover                         *
000000                  *                K. Koerth                              *
000000                  *                B. Kroll                               *
000000                  *                                                       *
000000                  *                IRT - Hannover  W. Gerth               *
000000                  *                                                       *
000000                  *       Date:     4.11.86       10:00:00               *
000000                  *       Version:     2.0        Atr: Colour            *
000000                  *-------------------------------------------------------*
000000                  *..... IMPLEMENTATION-DATA ......................*
000000 00000010         ACBS    EQU     16              ACIA-BUFFER-SIZE       *
000000 0000007F         ACRBM   EQU     $7F             READ-BUFFER-MASK       *
000000 0000007F         ACTBM   EQU     $7F             TRANSM. BUF MASK       *
000000 00FFFC00         ACIA1S  EQU     $FFFC00         AC1 STATUS  KEYBORD    *
000000 00FFFC02         ACIA1D  EQU     ACIA1S+2        DATA-ADR               *
000000 00FFFC04         ACIA3S  EQU     $FFFC04         AC2 STATUS  MIDI       *
000000 00FFFC06         ACIA3D  EQU     ACIA3S+2        DATA-ADR               *
000000 00FFFA01         MFP     EQU     $FFFA01         MFP 68901 START ADR    *
000000 00FFFA01         MGPIP   EQU     MFP             GENERAL PURPOSE I/O    *
000000 00FFFA03         MEAR    EQU     MFP+2           ACTIVE EDGE REG.       *
000000 00FFFA05         MDDR    EQU     MFP+4           DATA DIRECTION REG.    *
000000 00FFFA07         MIERA   EQU     MFP+6           INT. ENABLE REG. A     *
000000 00FFFA09         MIERB   EQU     MFP+8           INT. ENABLE REG. B     *
000000 00FFFA0B         MIPRA   EQU     MFP+10          INT. PENDING REG. A    *
000000 00FFFA0D         MIPRB   EQU     MFP+12          INT. PENDING REG. B    *
000000 00FFFA0F         MISRA   EQU     MFP+14          INT. IN-SERVICE R.A    *
000000 00FFFA11         MISRB   EQU     MFP+16          INT. IN-SERVICE R.B    *
000000 00FFFA13         MIMRA   EQU     MFP+18          INT. MASK REG. A       *
000000 00FFFA15         MIMRB   EQU     MFP+20          INT. MASK REG. B       *
000000 00FFFA17         MVR     EQU     MFP+22          VECTOR REG.            *
000000 00FFFA19         MTACR   EQU     MFP+24          TIMER A CONTROL REG    *
000000 00FFFA1B         MTBCR   EQU     MFP+26          TIMER B CONTROL REG    *
000000 00FFFA1D         MTCDCR  EQU     MFP+28          TIMER C+D CONTROL R    *
000000 00FFFA1F         MTADR   EQU     MFP+30          TIMER A DATA REG       *
000000 00FFFA21         MTBDR   EQU     MFP+32          TIMER B DATA REG       *
000000 00FFFA23         MTCDR   EQU     MFP+34          TIMER C DATA REG       *
000000 00FFFA25         MTDDR   EQU     MFP+36          TIMER D DATA REG       *
000000 00FFFA27         MSCR    EQU     MFP+38          SYNC CHARACTER REG     *
000000 00FFFA29         MUCR    EQU     MFP+40          USART CONTROL REG      *
000000 00FFFA2B         MRSR    EQU     MFP+42          RECIEVER STATUS REG    *
000000 00FFFA2D         MTSR    EQU     MFP+44          TRANSMIT STATUS REG    *
000000 00FFFA2F         MUDR    EQU     MFP+46          USART DATA REG         *
000000 00FF8800         PSGA    EQU     $FF8800         SOUNDCHIP ADR.-REG.    *
000000 00FF8802         PSGD    EQU     $FF8802         SOUNDCHIP DAT.-REG.    *
000000 00FF8604         DISK    EQU     $FF8604         DISK DATA REG.         *
000000 00FF8606         DMACR   EQU     $FF8606         DMA-CONTROLL-REG.      *
000000 0000000A         PPLDN   EQU     10              LDN OF PP-DRIVER       *
000000 00000070         IRVSIV  EQU     $70             VERTICAL-BLANK IID     *
000000 00000100         IRPAIV  EQU     $100            PARALLEL PORT IID      *
000000 00000114         IRCLIV  EQU     $114            CLOCK IID              *
000000 00000118         IRACIV  EQU     $118            ACIA IID               *
000000 00000128         IRTBIV  EQU     $128            TRANSMIT BUFFER IID    *
000000 00000130         IRRBIV  EQU     $130            RECEIVE BUFFER IID     *
000000 00000138         IRRIIV  EQU     $138            RING INDICATOR IID     *
000000                  *                                                       *
000000                  *-------------------------------------------------------*
000000                  *..... ACIA-PARAMTER INSIDE I6-BUFFER ...........*
000000 00000000                 DAR:     EQU    0       DATA REGISTER          *
```

```
000000 00000004          RSR:     EQU   DAR+4    RECIEVER STATUS REG *
000000 00000008          ACRB:    EQU   RSR+4    READ-BUFFER (CYCL)  *
000000 00000088          ACTB:    EQU   ACRB+ACRBM+1  TRANSM. BUFFER*
000000 00000108          ACRR:    EQU   ACTB+ACTBM+1  READER-READ PT*
000000 0000010A          ACRW:    EQU   ACRR+2          READER-WRIT PT*
000000 0000010C          ACTR:    EQU   ACRW+2  TRANSM. READ PTR    *
000000 0000010E          ACTW:    EQU   ACTR+2  TRANSM. WRITE PTR   *
000000 00000110          ACRTID:  EQU   ACTW+2  ACIA RECEIVER-TID   *
000000 00000114          ACTTID:  EQU   ACRTID+4 ACIA TRANSM.-TID   *
000000 00000118          ACTON:   EQU   ACTTID+4 XMITTER ON MASK    *
000000 00000119          ACIOFF:  EQU   ACTON+1 INPUT-SWITCH        *
000000 0000011A          ACIS:    EQU   ACTON+2 INPUT-SWITCH        *
000000 0000011C          ACRC:    EQU   ACTON+4 READ-COUNTER        *
000000 0000011E          ACTC:    EQU   ACRC+2  TRANSMIT-COUNTER    *
000000 00000120          ACSTR:   EQU   ACTC+2  STATUS-REGISTER     *
000000 00000121          ACEMU:   EQU   ACSTR+1 EMULATOR-FLAG       *
000000 00000122          ACRA1:   EQU   ACSTR+2 A1 (CE) FOR RECEIVER*
000000 00000126          ACTA1:   EQU   ACRA1+4 A1 (CE) FOR XMITTER *
000000 0000012A          ACCTID:  EQU   ACTA1+4 CMMD-TID            *
000000 0000012E          ACRCAD:  EQU   ACCTID+4  READER CE-TEXT AD *
000000 00000132          ACTCAD:  EQU   ACRCAD+4  TRANSM. CE-TEXT AD*
000000 00000136          ACLEN:   EQU   ACTCAD+4   ARRAY-LENGTH     *
000000 000003A2          ACIABU:  EQU   3*ACLEN   BUFFER SIZE ACIA  *
000000                 *-----------------------------------------------*
000000                 *     BITS IN ACSTR                             *
000000                 *                                               *
000000                 *     7        NO COMMANDS                      *
000000                 *     6                                         *
000000                 *     5                                         *
000000                 *     4        CLEAR IF TX IR PENDING           *
000000                 *     3        BINARY MODE                      *
000000                 *     2        MARK NEXT READER                 *
000000                 *     1        MODE (A/B/C)                     *
000000                 *     0        XMITTER CLEAR (XON/OFF)          *
000000                 *                                               *
000000                 *..... CENTRONIX-PORT AND FLOPPY DRIVE I6 BUF....*
000000 000003A2        STAPSG EQU    ACIABU          STATUS BYTE OF SOUND*
000000 000003A4        PPTID  EQU    STAPSG+2        TID OF TASK TO WAKE *
000000 000003A8        PPSTAT EQU    PPTID+4         SOFTWARE STATUS REG *
000000 000003AA        PPCAD  EQU    PPSTAT+2        CURRENT DATA ADR    *
000000 000003AE        PPCNT  EQU    PPCAD+4         BYTES STILL TO TX   *
000000 000003B2        CENLEN EQU    PPCNT+4         ARRAY-LENGTH        *
000000 000003B2        IRMODE EQU    CENLEN          MODE IN IR ROUTINE  *
000000 000003B3        IRSTAT EQU    IRMODE+1        ANSWER STATUS OF IR *
000000 000003B4        WSPADR EQU    IRSTAT+1        ADR OF TASK-WORKSP. *
000000 000003B8        FMTID  EQU    WSPADR+4        TID OF FILEMANAGER  *
000000 000003BC        TIMOUT EQU    FMTID+4         TIMEOUT COUNTER     *
000000 000003C0        OLDSP  EQU    TIMOUT+4        OLD STACK POINTER   *
000000 000003C4        I6BL   EQU    OLDSP+4         BUFFER-SIZE         *
000000                 *-----------------------------------------------*
000000                 *     L I V I N G   C O D E   ' I M P L '       *
000000                 *-----------------------------------------------*
000000                 RDRG   0                                         *
000000 000000000000
       00000010        DC     0,0,0,0,$0010                             *
00000A 496D70322E30    DC.8   'Imp2.0'                                  *
000010                 *..... HEAD-LINE OF IMPLEMENTATION ..............*
000010 AEB1BF9504C5    DC     $AEB1,$BF95,(16*2+1)*97                   *
000016 496D703D4174
       61726920322E
       30FF            DC.8   'Imp=Atari 2.0',$FF                       *
```

```
000024                  *-----------------------------------------------------*
000024                  *       DEVICE FACILITIES (SD/DD-OP.) INIT-TABLE      *
000024 AEB1BF950309             DC      $AEB1,$BF95,(10*2+1)*37               *
00002A 0000                     DC      $0000           START-LDN             *
00002C 3300                     DC      $3300           LDN 0 = ACIA1         *
00002E 0000                     DC      0                                     *
000030 AEB1BF950309             DC      $AEB1,$BF95,(10*2+1)*37               *
000036 0002                     DC      $0002           START LDN             *
000038 0B00                     DC      $0B00           LDN 2 = ACIA2 USART   *
00003A 0000                     DC      0                                     *
00003C AEB1BF950309             DC      $AEB1,$BF95,(10*2+1)*37               *
000042 0004                     DC      $0004           START LDN             *
000044 0300                     DC      $0300           LDN 4 = MIDI          *
000046 0000                     DC      0               STOP MARK             *
000048 AEB1BF950309             DC      $AEB1,$BF95,(10*2+1)*37               *
00004E 000A                     DC      $000A           START LDN             *
000050 2200                     DC      $2200           LDN10 = PARALL. PORT*
000052 0000                     DC      $0000           STOP-MASK             *
000054                  *-----------------------------------------------------*
000054                  *       DEVICE-MNEMO TO LDN/DRIVE-TABLE               *
000054 AEB1BF9502BF             DC      $AEB1,$BF95,(9*2+1)*37                *
00005A 41318000                 DC.B    'A1',$80,0      ACIA1                 *
00005E 41328200                 DC.B    'A2',$82,0      ACIA2  USART          *
000062 554C8203                 DC.B    'UL',$82,3      ACIA2-BUFF'D,NO LF    *
000066 42318002                 DC.B    'B1',$80,2      BUFFERED SCC1         *
00006A 42328202                 DC.B    'B2',$82,2      BUFFERED SCC2         *
00006E 43318006                 DC.B    'C1',$80,6      SCANNED SCC1          *
000072 43328206                 DC.B    'C2',$82,6      SCANNED SCC2          *
000076 50508A00                 DC.B    'PP',$8A,0      PARALLEL PORT         *
00007A 41338400                 DC.B    'A3',$84,0      MIDI                  *
00007E 42338402                 DC.B    'B3',$84,2      BUFFERED SCC3 MIDI    *
000082 43338406                 DC.B    'C3',$84,6      SCANNED SCC3 MIDI     *
000086 0000                     DC      0               END-MARKER            *
000088                  *  1.   REMOVE LAST DC 0 AND ADD DEV'S+LDN+DRIVE     *
000088                  *  2.   MAX. IDENTIFIER-LENGTH IS 6 BYTES            *
000088                  *  3.   THE LDN-BYTE MUST BE FLAGGED (+$80)          *
000088                  *       FLAG IS USED AS TEXT-LIMITER                 *
000088                  *  4.   CLOSE WITH    DC  0     ZERO-TEXT=END-MARK   *
000088                  *-----------------------------------------------------*
000088                  *       REQUEST INTERRUPT-BUFFER-SPACE                *
000088 AEB1BF95022B             DC      $AEB1,$BF95,(7*2+1)*37   IDP6         *
00008E 03C4                     DC      I6BL            REQU. SPACE  ACIA1-3*
000090                  *-----------------------------------------------------*
000090                  *       INSTALL TASK-HEADERS  SCC1/901/SCC3           *
000090 AEB1BF95006F             DC      $AEB1,$BF95,(1*2+1)*37 - TASK-INF.    *
000096 8180                     DC.B    $81,$80         RES. I/O              *
000098 23414349413 1            DC.B    '#ACIA1'        NAME                  *
00009E FFF8                     DC      -5              PRIO                  *
0000A0 000000F2                 DC.L    OPNAME+140      VTH                   *
0000A4 08EC                     DC      QACIA1-$        START-PC              *
0000A6 0000                     DC      0               LDN (CONSOLE)         *
0000A8                  *.....................................................*
0000A8 8180                     DC.B    $81,$80         RES. I/O              *
0000AA 235253323332             DC.B    '#RS232'        NAME                  *
0000B0 FFF8                     DC      -5              PRIO                  *
0000B2 000000F2                 DC.L    OPNAME+140      VTH                   *
0000B6 08EE                     DC      QACIA2-$        START-PC              *
0000B8 0002                     DC      2               LDN                   *
0000BA                  *.....................................................*
0000BA 8180                     DC.B    $81,$80         RES. I/O              *
0000BC 234D49444933             DC.B    '#MIDI3'        NAME                  *
```

```
0000C2 FFFB                 DC      -5              PRIO                 *
0000C4 000000F2             DC.L    OPNAME+140      VTH                  *
0000C8 0902                 DC      QACIA3-$        START-PC             *
0000CA 0004                 DC      4               LDN                  *
0000CC                *.......................................................*
0000CC 8180                 DC.B    $81,$80         RES. I/O             *
0000CE 2350504F5254         DC.B    '#PPORT'        NAME                 *
0000D4 FFFD                 DC      -3              PRIO                 *
0000D6 000000F2             DC.L    OPNAME+140      VTH                  *
0000DA 0C02                 DC      QCENT-$         START-PC             *
0000DC 000A                 DC      10              LDN                  *
0000DE                *......STOP-MARK ..................................*
0000DE 0000                 DC      0                                    *
0000E0                *********************************************************
0000E0                *       INITIALISATION-SEQUENCE                     *
0000E0                *-------------------------------------------------------*
0000E0 AEB1BF95047B         DC      $AEB1,$BF95,(15*2+1)*37   INIT-SEQ *
0000E6                *                                                     *
0000E6                *..... INITIALIZE TIMER  'C' SYSTEM CLOCK .......*
0000E6 13FC009A00FF
       FA23                 MOVE.B =$9A,MTCDR       TIMER C SET          *
0000EE                *..... INITIALIZE TIMER  'D' RS232 BAUDRATE .....*
0000EE 13FC000200FF
       FA25                 MOVE.B =$02,MTDDR       TIMER D SET 9600 Bd *
0000F6 13FC005100FF
       FA1D                 MOVE.B =$51,MTCDCR      TIMER D+C START      *
0000FE                *..... IF COLD START: SET VIDEO RAM ..............*
0000FE 4AB8081A             TST.L   VIDEOP          COLD START ?         *
000102 66000086             BNE     INIT1           B: IF NOT            *
000106 4281                 CLR.L   D1              USE TO CLEAR         *
000108 20380886             MOVE.L RAMEND,D0        RAM END ADRESS       *
00010C 2240                 MOVEA.L D0,A1           LOAD RAM END ADRESS *
00010E 048000007F00         SUBI.L =$7F00,D0        NEW SCREEN BEGIN     *
000114 4200                 CLR.B   D0              FORGET LAST BYTE     *
000116 2400                 MOVE.L D0,D2            SAVE SCREEN BEGIN    *
000118 04800000140E         SUBI.L =TEMUL+$E,D0     NEW MARK END         *
00011E 2040                 MOVEA.L D0,A0           LOAD MARK END ADRESS*
000120 20D1                 MOVE.L (A1),(A0)+       MAKE NEW MARK        *
000122 22C1                 MOVE.L D1,(A1)+         CLEAR OLD MARK       *
000124 2651                 MOVEA.L (A1),A3         ADRESS BACK-PTR.     *
000126 20D1                 MOVE.L (A1),(A0)+       SET BACK-PTR.        *
000128 22C1                 MOVE.L D1,(A1)+         CLEAR OLD MARK       *
00012A 2091                 MOVE.L (A1),(A0)        SET NEW MARK         *
00012C 2281                 MOVE.L D1,(A1)          CLEAR OLD MARK       *
00012E 2680                 MOVE.L D0,(A3)          STORE NEW FORW.-PTR.*
000130 21C00886             MOVE.L D0,RAMEND        STORE NEW RAM END    *
000134 21C00958             MOVE.L D0,SERTOP        SET SERTOP NEW       *
000138 21C2081A             MOVE.L D2,VIDEOP        STORE SCREEN BEGIN   *
00013C 263C00001400         MOVE.L =TEMUL,D3        EMUL STACK LENGTH    *
000142 9483                 SUB.L   D3,D2           IDP4 POINTER         *
000144 2042                 MOVEA.L D2,A0           ADRESS EMUL STACK    *
000146 2842                 MOVEA.L D2,A4           SAVE IDP4            *
000148 E28B                 LSR.L   =1,D3           LENGTH/2             *
00014A 4258           INIT0 CLR     (A0)+           CLEAR STACK          *
00014C 51CBFFFC             DBF     D3,INIT0        B: NOT READY         *
000150 397C00180020         MOVE    =24,SCRLEN(A4) SCREEN LENGTH = 25   *
000156 397C009F0022         MOVE    =$9F,SCRCO1(A4) COUNT FOR CLEAR     *
00015C 397C0F880024         MOVE    =$F88,SCRCO2(A4) COUNT SMOOTH SCROL*
000162 13FC000200FF
       820A                 MOVE.B =2,$FF820A       SYNC INTERN/50 Hz    *
00016A 33FCFFFF00FF
```

```
      8240                MOVE    =$FFFF,$FF8240 COLOUR PLANE 0        *
000172 33FC070000FF
      8242                MOVE    =$0700,$FF8242 PLANE 1               *
00017A 33FC077000FF
      8244                MOVE    =$0770,$FF8244 PLANE 2               *
000182 33FC000700FF
      8246                MOVE    =$0007,$FF8246 PLANE 3               *
00018A            *..... ENTRY FOR ABORT ..........................*
00018A 2438081A   INIT1 MOVE.L VIDEOP,D2        WARM START SET SCREE*
00018E 21C2083A         MOVE.L D2,IDP3          HIDE SCREEN-ADR.     *
000192 2202             MOVE.L D2,D1            SAVE POINTER         *
000194 E08A             LSR.L  =8,D2            SET VIDEO-CHIP:      *
000196 13C200FF8203     MOVE.B D2,$FF8203       LOW BWSP             *
00019C E08A             LSR.L  =8,D2            HIGH BWSP            *
00019E 13C200FF8201     MOVE.B D2,$FF8201       "                    *
0001A4 92BC00001400     SUB.L  =TEMUL,D1        TERMINAL-BUFFER PTR  *
0001AA 21C1083E         MOVE.L D1,IDP4          SET IDP4 NEW         *
0001AE D28C0000002C     ADD.L  =T4LEN,D1
0001B4 21C1081E         MOVE.L D1,FUNKP         FUNKTIONKEY-PTR.     *
0001B8 4288084A         CLR.L  IDP7             CLEAR NEW SCREEN-PTR*
0001BC            *..... CLEAR SCREEN .............................*
0001BC 2678083E   INIT2 MOVEA.L IDP4,A3         TERM.EMUL-POINTER    *
0001C0 42680004         CLR    STAT2(A3)        SET TO HIGH RESOLUT. *
0001C4 42680006         CLR    STAT3(A3)        CLEAR ESC-STATUS     *
0001C8 377C00030008     MOVE   =#3,COLOUR(A3)   SET COLOUR TO PAL.1  *
0001CE 2078081E         MOVEA.L FUNKP,A0        FUNKTIONKEY-PTR.     *
0001D2 D1FC00000320     ADDA.L =FKLEN,A0        CHAR.TABLE-PTR.      *
0001D8 22780836         MOVEA.L IDP2,A1         GET TABEL INDEX      *
0001DC 43E90018         LEA    ASC(A1),A1       CHAR SET FROM EPROM  *
0001E0 2449             MOVEA.L A1,A2           MAKE A COPY          *
0001E2 203C000002F8     MOVE.L =760,D0          COUNTER FOR 95 CHARS*
0001E8 0839000700FF
      FA01              BTST   =7,MFP           TEST MONITOR-TYP     *
0001F0 670E             BEQ.S  INIT3            B: IF HIGH-RES. MON  *
0001F2 08EB00030004     BSET   =HILOW,STAT2(A3) SET TO MIDDLE RES    *
0001F8 D3EA0004         ADDA.L 4(A2),A1         REAL CHAR SET        *
0001FC 303C017C         MOVE   =380,D0          COUNTER FOR 95 CHARS*
000200 D3D2       INIT3 ADDA.L (A2),A1          REAL CHAR SET        *
000202 30D9       INIT4 MOVE   (A1)+,(A0)+      CHAR-TAB TO RAM      *
000204 51C8FFFC         DBF    D0,INIT4         LOOPING              *
000208 20780836         MOVEA.L IDP2,A0         GET JUMPTABEL        *
00020C 4EA80000         JSR    CLRSCR(A0)       CLEAR SCREEN         *
000210 42680002         CLR    STAT1(A3)        CLEAR CONSOL-STATUS  *
000214 08EB00020002     BSET   =CAPSF,STAT1(A3) SET CAPSLOC          *
00021A 42680010         CLR    CHAR(A3)         CLEAR CHAR.-REPEAT   *
00021E 377C0016000E     MOVE   =#16,CURT(A3)    SET CURSOR BLINK DE. *
000224 377C001E0012     MOVE   =REP,AUTOD(A3)   SET DELAY RATE       *
00022A 377C00030014     MOVE   =RATE,AUTOS(A3)  SET STEP RATE        *
000230 42680016         CLR    KEYADR(A3)       CLEAR MAUS INPUT     *
000234 08EB00020004     BSET   =CURB,STAT2(A3)  CURSOR BLINK ON      *
00023A            *..... INITIALIZE  ACIAS   AND MFP 901 ..........*
00023A            *..... ACIA 1/2 SOFTWARE-STATUS-RESET ...........*
00023A 7003             MOVEQ  =$03,D0          FOR RAPID ACCESS     *
00023C 2278085E         MOVEA.L UITIDP,A1       USER-TID-POINTER     *
000240            *..... ACIA 1 ...................................*
000240 20780846         MOVEA.L IDP6,A0         IR-POINTER           *
000244 2159012A         MOVE.L (A1)+,ACCTID(A0)   STORE TID          *
000248 217C00FFFC02
      0000              MOVE.L =ACIA1D,DAR(A0) STORE ADR DATA REG *
000250 217C00FFFC00
      0004              MOVE.L =ACIA1S,RSR(A0) STORE RECIEVE  REG *
```

```
000258 5368011E          SUBQ    =1,ACTC(A0)      SET 'ALL DONE'      *
00025C 117C00010121      MOVE.B  =1,ACEMU(A0)     EMULATION ON        *
000262 45F900FFFC00      LEA     ACIA1S,A2        ACIA ADR            *
000268 1480              MOVE.B  D0,(A2)          MASTER-RESET        *
00026A 7296              MOVEQ   =$96,D1          READY TO READ       *
00026C 1481              MOVE.B  D1,(A2)          READY TO READ       *
00026E 11410119          MOVE.B  D1,ACTOFF(A0)    SET OFF CONTROL BYTE*
000272 117C00B60118      MOVE.B  =$B6,ACTON(A0)   SET ON  CONTROL BYTE*
000278                 *..... ACIA 2 = USART 901.......................*
00027A 41E80136          LEA     ACLEN(A0),A0     IR POINTER          *
00027C 2159012A          MOVE.L  (A1)+,ACCTID(A0) STORE TID           *
000280 217C00FFFA2F
       0000              MOVE.L  =MUDR,DAR(A0)    STORE DATA REG ADR  *
000288 217C00FFFA28
       0004              MOVE.L  =MRSR,RSR(A0)    STORE RECIEVE REG   *
000290 5368011E          SUBQ    =1,ACTC(A0)      SET 'ALL DONE'      *
000294 13FC009800FF
       FA27              MOVE.B  =$98,MUCR        /16, 8BITS, 1START  *
00029C                 *                          2STOP, NO PARITY    *
00029C 13FC000100FF
       FA2D              MOVE.B  =1,MTSR          TRANSMIT ENABLE     *
0002A4 13FC000100FF
       FA28              MOVE.B  =1,MRSR          RECIEVER ENABLE     *
0002AC                 *......ACIA 3 = MIDI............................*
0002AC 41E80136          LEA     ACLEN(A0),A0     IR PIONTER          *
0002B0 2159012A          MOVE.L  (A1)+,ACCTID(A0) STORE TID           *
0002B4 217C00FFFC06
       0000              MOVE.L  =ACIA3D,DAR(A0) STORE ADR DATA REG   *
0002BC 217C00FFFC04
       0004              MOVE.L  =ACIA3S,RSR(A0) STORE RECIEVE  REG   *
0002C4 5368011E          SUBQ    =1,ACTC(A0)      SET 'ALL DONE'      *
0002C8 45F900FFFC04      LEA     ACIA3S,A2        ACIA ADR            *
0002CE 1480              MOVE.B  D0,(A2)          MASTER-RESET        *
0002D0 7295              MOVEQ   =$95,D1          READY TO READ       *
0002D2 1481              MOVE.B  D1,(A2)          READY TO READ       *
0002D4 11410119          MOVE.B  D1,ACTOFF(A0)    SET OFF CONTROL BYTE*
0002D8 117C00B50118      MOVE.B  =$B5,ACTON(A0)   SET ON  CONTROL BYTE*
0002DE                 *..... INIT MFP 68901 ..........................*
0002DE 13FC004000FF
       FA17              MOVE.B  =$40,MVR         SET VECTOR REGISTER  *
0002E6 13FC000000FF
       FA03              MOVE.B  =$0,MEAR         USE FALLING EDGE     *
0002EE 13FC000000FF
       FA05              MOVE.B  =$0,MDDR         USE AS INPUT         *
0002F6 13FC005400FF
       FA07              MOVE.B  =$54,MIERA       ENABLE INT. A        *
0002FE 13FC00E100FF
       FA09              MOVE.B  =$E1,MIERB       ENABLE INT. B        *
000306 13FC005400FF
       FA13              MOVE.B  =$54,MIMRA       SET INT. MASK A      *
00030E 13FC00E100FF
       FA15              MOVE.B  =$E1,MIMRB       SET INT. MASK B      *
000316                 *..... DUMMY OUTPUT MFP 901.....................*
000316 13FC000000FF
       FA2F              MOVE.B  =0,MUDR          DUMMY OUTPUT         *
00031E 4A3900FFFA2F      TST.B   MUDR             DUMMY  INPUT         *
000324                 *......SEARCH THE C'T CLOCK .....................*
000324 203C0000FF9C      MOVE.L  =$0FF9C,D0       SELECT TAKT          *
00032A 223C0000FFFC      MOVE.L  =$0FFFC,D1          "                 *
000330 243C0000FFC0      MOVE.L  =$0FFC0,D2       READ REGISTER        *
000336 263C0000FFFF      MOVE.L  =$0000FFFF,D3    COUNTER              *
```

```
00033C 283C0000FFEE          MOVE.L =$0FFEE,D4       SELECT CLOCK          *
000342 43F900FB0000          LEA    $FB0000,A1       1ST ROM ADR. EVEN     *
000348 611A          SERCL1  BSR.S  SCLOCK           SEARCH THE CLOCK      *
00034A 6768                  BEQ.S  READ             B: IF FOUND           *
00034C D3FC00000001          ADDA.L =1,A1            NEXT ROM ODD          *
000352 6110                  BSR.S  SCLOCK           SEARCH THE CLOCK      *
000354 675E                  BEQ.S  READ             B: IF FOUND           *
000356 D3C3                  ADDA.L D3,A1            NEXT EVEN ROM         *
000358 B3FC00FF0000          CMPA.L =$FF0000,A1      TEST IF READY         *
00035E 66E8                  BNE.S  SERCL1           B: NEXT ROUND         *
000360 600000DC              BRA    ISOUND           B: NO CLOCK FOUND     *
000364 3E3C0064      SCLOCK  MOVE   =100,D7          LOOP COUNTER          *
000368 1C314800              MOVE.B 0(A1,D4.L),D6    SELECT THE CLOCK      *
00036C 1C314800              MOVE.B 0(A1,D4.L),D6    SELECT THE CLOCK      *
000370 1C314800              MOVE.B 0(A1,D4.L),D6    SELECT THE CLOCK      *
000374 1C310800              MOVE.B 0(A1,D0.L),D6    SELECT REGISTER       *
000378 1C311800              MOVE.B 0(A1,D1.L),D6    SELECT REGISTER       *
00037C 1C312800              MOVE.B 0(A1,D2.L),D6    READ REGISTER         *
000380 1C312800              MOVE.B 0(A1,D2.L),D6    READ REGISTER         *
000384 1A06                  MOVE.B D6,D5            MAKE A COPY           *
000386 BA06          SCLO1   CMP.B  D6,D5            TEST IF STABLE        *
000388 6626                  BNE.S  SCLO3            B: CLOCK FOUND        *
00038A 1A06                  MOVE.B D6,D5            MAKE A COPY           *
00038C 1C314800              MOVE.B 0(A1,D4.L),D6    SELECT THE CLOCK      *
000390 1C314800              MOVE.B 0(A1,D4.L),D6    SELECT THE CLOCK      *
000394 1C314800              MOVE.B 0(A1,D4.L),D6    SELECT THE CLOCK      *
000398 1C310800              MOVE.B 0(A1,D0.L),D6    SELECT REGISTER       *
00039C 1C311800              MOVE.B 0(A1,D1.L),D6    SELECT REGISTER       *
0003A0 1C312800              MOVE.B 0(A1,D2.L),D6    READ REGISTER         *
0003A4 1C312800              MOVE.B 0(A1,D2.L),D6    READ REGISTER         *
0003A8 51CFFFDC              DBF    D7,SCLO1         LOOP                  *
0003AC 7E01                  MOVEQ  =1,D7            RETURNS WITH NE       *
0003AE 4E75                  RTS                                           *
0003B0 7E00          SCLO3   MOVEQ  =0,D7            RETURNS WITH EQ       *
0003B2 4E75                  RTS                                           *
0003B4               *.....  READ THE C'T CLOCK ......................*
0003B4 487A0072      READ    PEA    READ4            RETURN ADR.           *
0003B8 D3FC0000FF00          ADDA.L =$FF00,A1        RTC 58321 BASIC-ADR.*
0003BE 2449                  MOVEA.L A1,A2           MAKE A COPY           *
0003C0 2649                  MOVEA.L A1,A3           MAKE A COPY           *
0003C2 2849                  MOVEA.L A1,A4           MAKE A COPY           *
0003C4 D5FC00000080          ADDA.L =$80,A2          READ ENABLE           *
0003CA D7FC000000E0          ADDA.L =$E0,A3          READ LATCH            *
0003D0 99FC00000100          SUBA.L =$100,A4         DESELECT CLOCK        *
0003D6 4282                  CLR.L  D2                                     *
0003D8 4283                  CLR.L  D3               CLEAR REGISTER        *
0003DA 7205                  MOVEQ  =5,D1            START WITH H10        *
0003DC 7E03                  MOVEQ  =3,D7            INITIAL MASK H10      *
0003DE 610E          READ1   BSR.S  READ2            READ CLOCK TO D0      *
0003E0 16381041              MOVE.B READ3+1(D1.W),D3  FACTOR               *
0003E4 C4C3                  MULU   D3,D2            *10 OR *6             *
0003E6 D480                  ADD.L  D0,D2            ACCUMULATE            *
0003E8 4A41                  TST    D1               IF ALL DONE           *
0003EA 6CF2                  BGE.S  READ1            B: CONTINUE           *
0003EC 4E75                  RTS                                           *
0003EE               *.......................................................*
0003EE 3A01          READ2   MOVE   D1,D5            MAKE A COPY           *
0003F0 E345                  ASL    =1,D5            SET TO EVEN           *
0003F2 1C2900EE              MOVE.B $EE(A1),D6       SELECT CLOCK          *
0003F6 1C2900EE              MOVE.B $EE(A1),D6         "      "            *
0003FA 1C2900EE              MOVE.B $EE(A1),D6         "      "            *
```

```
0003FE 10325000                MOVE.B 0(A2,D5),D0     SELECT REG.          *
000402 10335000                MOVE.B 0(A3,D5),D0                          *
000406 102900C0                MOVE.B $C0(A1),D0      READ THE CLOCK       *
00040A 102900C0                MOVE.B $C0(A1),D0      THE NIBBLE IS IN D0  *
00040E 182A001E                MOVE.B $1E(A2),D4      SAVE THE CLOCKTIME   *
000412 182900EE                MOVE.B $EE(A1),D4                           *
000416 1C14                    MOVE.B (A4),D6         DESELECT CLOCK       *
000418 4600                    NOT.B  D0              DATA IN D0           *
00041A C087                    AND.L  D7,D0           MASK                 *
00041C 7E0F                    MOVEQ  =$0F,D7         MASK NEXT READ       *
00041E 5341                    SUBQ   =1,D1           ADR. REDUCTION       *
000420 4E75                    RTS                                         *
000422                *.......................................................*
000422 0A060A060A01 READ3      DC.B   10,6,10,6,10,1 FACTOR TABLE          *
000428                *.......................................................*
000428 7002         READ4      MOVEQ  =2,D0           LOOP 3 TIMES         *
00042A E38A         READ5      LSL.L  =1,D2           *2                   *
00042C 2E02                    MOVE.L D2,D7           SAVE *2              *
00042E E58A                    LSL.L  =2,D2           *8                   *
000430 D487                    ADD.L  D7,D2           10*D2                *
000432 51C8FFF6                DBF    D0,READ5        LOOPING              *
000436 94B8088E                SUB.L  $88E,D2         TIME BASE            *
00043A 21C2088A                MOVE.L D2,$88A         SET THE CLOCK        *
00043E                *..... INIT SOUND CHIP ..........................*
00043E 20780846     ISOUND     MOVEA.L IDP6,A0        IR POINTER           *
000442 317CE70003A2            MOVE   =$E700,STAPSG(A0) SET SOUND STATUS   *
000448 33FC000700FF
       8800                    MOVE   =$07,PSGA       SELECT I/O ENABLE    *
000450 33FC00C000FF
       8802                    MOVE   =$C0,PSGD       PORT A,B OUTPUT      *
000458 13FC000E00FF
       8800                    MOVE.B =$0E,PSGA       SELECT PORT A        *
000460 13E803A200FF
       8802                    MOVE.B STAPSG(A0),PSGD SEND THE MODE BYTE   *
000468 47FA0008                LEA    IMPEX,A3        RETURN ADR           *
00046C 4240                    CLR    D0              SEND DATA 00         *
00046E 60000836                BRA    PPOUT           DUMMY OUTPUT         *
000472 4E75         IMPEX      RTS                    EXIT OF INIT SEQ.    *
000474                *-----------------------------------------------------*
000474                *..... EXCEPTION-LINK-TABLE    THIS IMP .........*
000474 AEB1BF950431            DC     $AEB1,$BF95,(14*2+1)*37              *
00047A 006800D4                DC     $68,IR668-$-2       HORIZ. BLANKING  *
00047E 007000DA                DC     $70,IR670-$-2       VERTI. BLANKING  *
000482 0100001A                DC     $100,IR6100-$-2     PARALLEL PORT    *
000486 01180092                DC     $118,IR6118-$-2     ACIA1/2DATA-IR   *
00048A 012801D2                DC     $128,IR6128-$-2     901 TRANSM B E   *
00048E 013001EE                DC     $130,IR6130-$-2     901 RECIEV B F   *
000492 011404C6                DC     $114,IR6114-$-2     CLOCK-IR         *
000496 0138005A                DC     $138,IR6138-$-2     RING INDICATOR   *
00049A 0000                    DC     0               *** END OF EXCTBL    *
00049C                *-----------------------------------------------------*
00049C                *......PARALLEL-PORT-INTERRUPT ..................*
00049C 04A2                    DC     IR6Z01-IR6100   MALFUNC-ADR          *
00049E 3F3807FE     IR6100     MOVE   IID,-(A7)       SAVE INTER.ID        *
0004A2 31FC010007FE            MOVE   =IRPAIV,IID     NEW VECTOR ADR.      *
0004A8 48E7C0F8                MOVEM.L D0/D1/A0/A1/A2/A3/A4,-(A7) SAVE     *
0004AC 20780846                MOVEA.L IDP6,A0        BUFFER-POINTER       *
0004B0 4A6803AE                TST    PPCNT(A0)       INSPECT COUNTER      *
0004B4 6730                    BEQ.S  IR61A2          B:NO JOB  DISABLE    *
0004B6 226803AA                MOVEA.L PPCAD(A0),A1   DATA-ADR             *
0004BA 47FA0008                LEA    IR61A1,A3       RETURN ADR           *
```

```
0004BE 1019                 MOVE.B (A1)+,D0        GET THE BYTE          *
0004C0 600007E4             BRA    PPOUT           TRANSMIT A BYTE       *
0004C4 214903AA      IR61A1 MOVE.L A1,PPCAD(A0)    STORE FOR NEXT READ   *
0004C8 536803AE             SUBQ   =1,PPCNT(A0)    COUNTER-REDUCTION     *
0004CC 6E1E                 BGT.S  IR61A3          B:DIRECT EXIT         *
0004CE 226803A4             MOVEA.L PPTID(A0),A1   FETCH TID             *
0004D2 B3FC00000000         CMPA.L =$0,A1          IS TID SET?           *
0004D8 670C                 BEQ.S  IR61A2          IF NOT                *
0004DA 08A900040024         BCLR   =BLKBSU,BLOCK(A1)  RUNNABLE           *
0004E0 53780800             SUBQ   =1,DPC          ALARM DISP.           *
0004E4 6006                 BRA.S  IR61A3          AND EXIT              *
0004E6 08E8000003A8  IR61A2 BSET   =0,PPSTAT(A0)   THE PORT IS FREE      *
0004EC 60000452      IR61A3 BRA    IR6Z01          EXIT                  *
0004F0               *-----------------------------------------------------*
0004F0               *..... RING INDICATOR ..............................*
0004F0 001A                 DC     IRRMF-IR6138    MALFUNC-ADR.          *
0004F2 3F3807FE      IR6138 MOVE   IID,-(A7)       SAVE IID              *
0004F6 31FC013807FE         MOVE   =IRRIIV,IID     STORE NEW INTERR.     *
0004FC 48E74340             MOVEM.L D1/D6/D7/A1,-(A7) SAVE               *
000500 223C20000000         MOVE.L =$20000000,D1   SET EVENT CODE        *
000506 2278080E             MOVEA.L PIRTRI,A1      SOFTWARE INTR.        *
00050A 4ED1                 JMP    (A1)            AND EXIT              *
00050C               *..... MALFUNKTION PROCESS ......................*
00050C 4CDF02C2      IRRMF  MOVEM.L (A7)+,D1/D6/D7/A1                    *
000510 31DF07FE             MOVE   (A7)+,IID       RELOAD IID            *
000514 60000432             BRA    DISP            EXIT                  *
000518               *-----------------------------------------------------*
000518               *..... N I G H T M A R E : A C I A  1/2/3 IR ....*
000518 041A                 DC     IR6Z00-IR6118   MALFUNC-ADR           *
00051A 3F3807FE      IR6118 MOVE   IID,-(A7)       SAVE IID              *
00051E 31FC011807FE         MOVE   =IRACIV,IID     NEW VECTOR ADR.       *
000524 48E7C0F8             MOVEM.L D0/D1/A0/A1/A2/A3/A4,-(A7)  SAVE     *
000528 49FA040A      IR611X LEA    IR6Z00,A4       QUICK EXIT            *
00052C 20780846             MOVEA.L IDP6,A0        BUFFER INDEX ACIA1    *
000530 0839000700FF
       FC00                 BTST   =7,ACIA1S       INTERRUPT REQUEST ?   *
000538 66000108             BNE    IR6P00          B IF YES              *
00053C 41E8026C             LEA  2*ACLEN(A0),A0    GET MIDI BUFFER       *
000540 0839000700FF
       FC04                 BTST   =7,ACIA3S       INTERRUPT REQUEST ?   *
000548 660000F8             BNE    IR6P00          B IF YES              *
00054C 600003F2             BRA    IR6Z01          EXIT                  *
000550               *                                                     *
000550               *..... H-SYNC. ......................................*
000550 00570300      IR668  ORI    =$0300,(A7)     SET IR-LEVEL 3        *
000554 600003F2             BRA    DISP                                  *
000558               *                                                     *
000558               *..... V-SYNC. ......................................*
000558 03E6                 DC     IR6Z01-IR670    MALFUNC-ADR           *
00055A 3F3807FE      IR670  MOVE   IID,-(A7)       SAVE IID              *
00055E 31FC007007FE         MOVE   =IRVSIV,IID     STORE NEW INTERR.     *
000564 48E7C0F8             MOVEM.L D0/D1/A0/A1/A2/A3/A4,-(A7)  SAVE     *
000568 49FA03D6             LEA    IR6Z01,A4       QUICK EXIT            *
00056C 26780846             MOVEA.L IDP6,A3        BUFFER INDEX          *
000570 4A2B03B2             TST.B  IRMODE(A3)      FLOPPY ACTIVE ?       *
000574 6722                 BEQ.S  IR67A           B: NOT ACTIVE         *
000576 53AB03BC             SUBQ.L =1,TIMOUT(A3)   DECREMENT TIMEOUT-C.  *
00057A 6A1C                 BPL.S  IR67A           B: SOME TIME LEFT     *
00057C 177C00FF03B2         MOVE.B =$FF,IRMODE(A3) SET FLAG TIMEOUT      *
000582 33FC008000FF
       8606                 MOVE   =$0080,DMACR    SELECT FLOPPY COMAND*
```

```
00058A 7040                  MOVEQ  =$40,D0          COUNTER FOR WAIT     *
00058C 51C8FFFE              DBF    D0,$             WAIT                 *
000590 33FC00D800FF
       8604                  MOVE   =$00D8,DISK      ABORT COMAND         *
000598 2678083E      IR67A   MOVEA.L IDP4,A3         BUFFER INDEX         *
00059C 2038084A              MOVE.L IDP7,D0          SCREEN CHANGE ?      *
0005A0 6710                  BEQ.S  IR671            B:NO CHANGE          *
0005A2 E088                  LSR.L  =8,D0            SET VIDEO CHIP       *
0005A4 13C000FF8203          MOVE.B D0,$FF8203       LOW BWSP             *
0005AA E088                  LSR.L  =8,D0                                 *
0005AC 13C000FF8201          MOVE.B D0,$FF8201       HIGH BWSP            *
0005B2 082B00020004  IR671   BTST   =CURB,STAT2(A3) CURSOR BLINK OK ?    *
0005B8 6726                  BEQ.S  IR672            B: IF NO BLINK       *
0005BA 082B00070002          BTST   =ACTION,STAT1(A3) CHAR IN SERVICE ?*
0005C0 661E                  BNE.S  IR672            B: YES               *
0005C2 536B000E              SUBQ   =1,CURT(A3)      INT.COUNTER-1        *
0005C6 6A18                  BPL.S  IR672            B: NOT READY TO BLIN*
0005C8 377C0016000E          MOVE   =$16,CURT(A3)    INT.COUNTER =$16     *
0005CE 2002                  MOVE.L D2,D0            SAVE D2              *
0005D0 20780836              MOVEA.L IDP2,A0         GET JUMPTABEL        *
0005D4 4EA80004              JSR    CHAINV(A0)       CURSOR BLINK         *
0005D8 086B00060002          BCHG   =CURSF,STAT1(A3) CURSOR ON/OFF        *
0005DE 2400                  MOVE.L D0,D2            RELOAD D2            *
0005E0               *.....  AUTOREPEAT ..............................*
0005E0 302B0010      IR672   MOVE   CHAR(A3),D0      READ FOR AUTOREP.    *
0005E4 673E                  BEQ.S  IR676            B: NO AUTOREPEAT     *
0005E6 082B00000004          BTST   =AUTOY,STAT2(A3) AUTOREP FREE ?       *
0005EC 6614                  BNE.S  IR673            B: IF YES            *
0005EE 322B0012              MOVE   AUTOD(A3),D1     LOAD DELAY COUNTER   *
0005F2 51C90026              DBF    D1,IR674         B: NOT DELAY END     *
0005F6 08EB00000004          BSET   =AUTOY,STAT2(A3) AUTOREP FREE !       *
0005FC 377C001E0012          MOVE   =REP,AUTOD(A3)   SET COUNTER NEW      *
000602 322B0014      IR673   MOVE   AUTOS(A3),D1     REPEAT COUNTER       *
000606 51C90018              DBF    D1,IR675         EXIT                 *
00060A 377C00030014          MOVE   =RATE,AUTOS(A3) SET COUNTER NEW       *
000610 20780846              MOVEA.L IDP6,A0         BUFFER-INDEX         *
000614 1200                  MOVE.B D0,D1            DUMMY MOVE           *
000616 60000866              BRA    OUT              DO AUTOREPEAT        *
00061A 37410012      IR674   MOVE   D1,AUTOD(A3)     SAVE DELAY COUNTER   *
00061E 6004                  BRA.S  IR676                                 *
000620 37410014      IR675   MOVE   D1,AUTOS(A3)     SAVE STEP COUNTER    *
000624 083800060822  IR676   BTST   =6,EVMASK        INT ENABLE ?         *
00062A 6714                  BEQ.S  IR677            B: NOT FREE          *
00062C 4CDF1F03              MOVEM.L (A7)+,D0/D1/A0/A1/A2/A3/A4 RELOAD *
000630 48E74340              MOVEM.L D1/D6/D7/A1,-(A7) SAVE               *
000634 223C40000000          MOVE.L =$40000000,D1    SET EVENT CODE       *
00063A 2278080E              MOVEA.L PIRTRI,A1       SOFTWARE INTR        *
00063E 4ED1                  JMP    (A1)             AND EXIT             *
000640 4ED4          IR677   JMP    (A4)             QUICK  EXIT          *
000642               *                                                    *
000642               *.....  CALLER DETECTED. DISPATCH IN/OUT .........*
000642 26680004      IR6P00  MOVEA.L RSR(A0),A3      GET ADR STATUS REG   *
000646 08130000              BTST   =0,(A3)          TEST IF READER CALL *
00064A 670000FC              BEQ    IR6T00           B: GO OUTPUT SECTION*
00064E 4A280121              TST.B  ACEMU(A0)        EMULATION ON ?       *
000652 67000044              BEQ    IR6I00           B: NO                *
000656 4240                  CLR.W  D0                                    *
000658 60000716              BRA    KEYINA           GET ASCII-STRING D0 *
00065C               *                               B: IR6I02            *
00065C               *.....  TRANSMITTER-BUFFER EMPTY ................*
00065C 02D6                  DC     IR6Z00-IR6128    MALFUNC-ADR          *
```

```
00065E 3F3807FE      IR6128 MOVE    IID,-(A7)       SAVE IID                *
000662 31FC012807FE         MOVE    =IRTBIV,IID     STORE NEW IID           *
000668 48E7C0F8             MOVEM.L D0/D1/A0/A1/A2/A3/A4,-(A7)  SAVE        *
00066C 49FA02C6             LEA     IR6Z00,A4       QUICK EXIT              *
000670 20780846             MOVEA.L IDP6,A0         BUFFER-INDEX            *
000674 41E80136             LEA     ACLEN(A0),A0    GET USART BUFFER-I      *
000678 600000CE             BRA     IR6T00          GO OUTPUT-SECTION       *
00067C               *                                                       *
00067C               *..... RECIEVER-BUFFER FULL ......................*
00067C 02B6                 DC      IR6Z00-IR6130   MALFUNC-ADR             *
00067E 3F3807FE      IR6130 MOVE    IID,-(A7)       SAVE IID                *
000682 31FC013007FE         MOVE    =IRRBIV,IID     STORE NEW IID           *
000688 48E7C0F8             MOVEM.L D0/D1/A0/A1/A2/A3/A4,-(A7)  SAVE        *
00068C 49FA02A6             LEA     IR6Z00,A4       QUICK EXIT              *
000690 20780846             MOVEA.L IDP6,A0         BUFFER-INDEX            *
000694 41E80136             LEA     ACLEN(A0),A0    GET USART BUFFER-I      *
000698               *..... RECEIVER-CALL ............................*
000698 26680004      IR6I00 MOVEA.L RSR(A0),A3      GET ADR RECIEV STATU*
00069C 707F                 MOVEQ   =$7F,D0         MASK, NO PARITY         *
00069E 082800030120         BTST    =3,ACSTR(A0)    BINARY MODE ?           *
0006A4 6704                 BEQ.S    IR6II0         B: NO BIN-MODE          *
0006A6 303CFFFF             MOVE    =$FFFF,D0       NO MASK ON DATA         *
0006AA 08130003      IR6II0 BTST    =3,(A3)         TEST BREAK              *
0006AE 6706                 BEQ.S   IR6I01          B:NO BREAK              *
0006B0               *..... BREAK-TEST ...............................*
0006B0 4240                 CLR     D0              FOR 'BREAK'             *
0006B2 60000112             BRA     IR6U00          UPDATE CMMD             *
0006B6               *....  REAL DATA-READ ...........................*
0006B6 26680000      IR6I01 MOVEA.L DAR(A0),A3      GET ADR DATA            *
0006BA C013                 AND.B   (A3),D0         DATA                    *
0006BC 1200          IR6I02 MOVE.B D0,D1            MOVE DATA               *
0006BE 6736                 BEQ.S   IR6I04          IGNORE 'NUL'            *
0006C0 4A280120             TST.B   ACSTR(A0)       NEG IF 'NO COMMANDS'*
0006C4 6B08                 BMI.S   IR6I03          B:COMMANDS INHIBITED*
0006C6 B23C0003             CMP.B   =3,D1           TEST 1,2,3              *
0006CA 6F0000FA             BLE     IR6U00          B:UPDATE CMMD           *
0006CE 923C0011      IR6I03 SUB.B   =$11,D1         TEST ON 'X-ON'          *
0006D2 6624                 BNE.S   IR6I05          B:IS NOT 'X-ON'         *
0006D4               *..... PROCESS 'X-ON'-COMMAND  ..................*
0006D4 08A800000120         BCLR    =0,ACSTR(A0)    FLAG XMITTER CLEAR      *
0006DA 082800040120         BTST    =4,ACSTR(A0)    CLEAR IF IR PENDING *
0006E0 6614                 BNE.S   IR6I04          B:TX ACTIVE, EXIT       *
0006E2               *......AWAKE SLEEPING TX IF NESESSARY............*
0006E2 3028010C             MOVE    ACTR(A0),D0     XMITTER READ PTR        *
0006E6 B068010E             CMP     ACTW(A0),D0     ANY CHAR TO WRITE ? *
0006EA 66000066             BNE     IR6T01          B:YES MAKE OUTPUT       *
0006EE 3028011E             MOVE    ACTC(A0),D0     TRANSMITTER COUNTER *
0006F2 6E000088             BGT     IR6T03          B:OUTPUT FROM CE        *
0006F6 4ED4          IR6I04 JMP     (A4)            EXIT                    *
0006F8               *..... TEST FOR X-OFF ...........................*
0006F8 5501          IR6I05 SUBQ.B =$13-$11,D1      TEST ON $13 X-OFF       *
0006FA 6608                 BNE.S   IR6I06          B:NOT X-OFF/X-ON        *
0006FC               *..... PROCESS X-OFF ............................*
0006FC 08E800000120         BSET    =0,ACSTR(A0)    FLAG XMITTER-STOP       *
000702 4ED4                 JMP     (A4)            INTERRUPT-EXIT          *
000704               *..... PROCESS NORMAL INPUT-CHARACTER               *
000704 244C          IR6I06 MOVEA.L A4,A2           RETURN = IR-RETURN      *
000706 4A28011A             TST.B   ACIS(A0)        INPUT-SWITCH            *
00070A 6708                 BEQ.S   IR6I07          B:PUT INTO CYC-BUF      *
00070C               *..... INPUT INTO CE ............................*
00070C 22680122             MOVEA.L ACRA1(A0),A1    LOAD A1                 *
```

```
000710 60000170              BRA    IR6Y00          STORE DO IN CE+IEXIT*
000714                 *..... PUT DATA INTO CYCLIC INPUT-BUFFER        *
000714 3228010A        IR6I07 MOVE   ACRW(A0),D1     WRITE-POINTER     *
000718 11801008              MOVE.B DO,ACRB(A0,D1.W)  STORE INFORM.   *
00071C 5241                  ADDQ   =1,D1          MOVE POINTER      *
00071E C27C007F              AND    =ACRBM,D1      READ-BUFFER-MASK  *
000722 B2680108              CMP    ACRR(A0),D1    IF READER REACHED *
000726 6704                  BEQ.S  IR6I08         B:DO NOT STORE    *
000728 3141010A              MOVE   D1,ACRW(A0)    NEW WRITE-POINTER *
00072C                 *..... STOP OUTSIDE XMITTER IF ON 'B'-DEVICE   *
00072C 4A280121        IR6I08 TST.B  ACEMU(A0)      IF EMUL. TERMINAL? *
000730 6614                  BNE.S  IR6I09         B:YES, NO ACTION  *
000732 082800010120          BTST   =1,ACSTR(A0)   IS SET IF LAST WAS B*
000738 670C                  BEQ.S  IR6I09         B: NO ACTION      *
00073A 08E800020120          BSET   =2,ACSTR(A0)   MARK FOR NEXT READER*
000740 7013                  MOVEQ  =$13,D0        CHAR TO SEND      *
000742 600000CE              BRA    IR6X00         XMIT BY FIRST SERV. *
000746 4ED4            IR6I09 JMP    (A4)           IR-EXIT           *
000748                 *-----------------------------------------------*
000748                 *..... TRANSMITTER-CALL ......................*
000748 3028010C        IR6T00 MOVE   ACTR(A0),D0     TRANSMITTER READ PTR*
00074C B068010E              CMP    ACTW(A0),D0    TEST EMPTY BUFFER *
000750 6722                  BEQ.S  IR6T02         B:BUFFER EMPTY    *
000752                 *..... FIRST SERVICE FROM CYCLIC BUFFER .......*
000752 08E800040120 IR6T01 BSET    =4,ACSTR(A0)   FLAG XMITTER ACTIVE *
000758 26680000              MOVEA.L DAR(A0),A3    GET ADR DATA      *
00075C D0FC0088              ADDA   =ACTB,A0       CYC.-BUFFER PTR   *
000760 16800000              MOVE.B O(A0,D0.W),(A3) WRITE            *
000764 90FC0088              SUBA   =ACTB,A0       OLD PTR           *
000768 5240                  ADDQ   =1,D0          MOVE READER UP    *
00076A C07C007F              AND    =ACTBM,D0      ACIA TRANSM.BUF MASK*
00076E 3140010C              MOVE   D0,ACTR(A0)    STORE NEW VALUE   *
000772 4ED4                  JMP    (A4)           IR-EXIT           *
000774                 *..... SERVICE ON COMMUNICATION-ELEMENT  ..... *
000774 3028011E        IR6T02 MOVE   ACTC(A0),D0     TRANSMITTER-COUNT *
000778 6724                  BEQ.S  IR6T04         B:ELEMENT READY   *
00077A 6B34                  BMI.S  IR6T05         ALLREADY ALL DONE *
00077C                 *..... REAL SERVICE ON COMMUNICATION ELEMENT   *
00077C 082800000120 IR6T03 BTST    =0,ACSTR(A0)   TEST IF XMITTER 'ON'*
000782 662C                  BNE.S  IR6T05         B:XMITTER-STOP    *
000784 24680132              MOVEA.L ACTCAD(A0),A2 TRANSMITTER ADR   *
000788 08E800040120          BSET   =4,ACSTR(A0)   FLAG XMITTER ACTIVE *
00078E 26680000              MOVEA.L DAR(A0),A3    GET ADR DATA      *
000792 169A                  MOVE.B (A2)+,(A3)     WRITE             *
000794 214A0132              MOVE.L A2,ACTCAD(A0)  MOVED POINTER     *
000798 5368011E              SUBQ   =1,ACTC(A0)    TRANSM COUNTER    *
00079C 4ED4                  JMP    (A4)           IR-EXIT           *
00079E 24680114        IR6T04 MOVEA.L ACTTID(A0),A2 ACIA TRANSM. TID *
0007A2 08AA00040024          BCLR   =BLKBSU,BLOCK(A2)  CONTINUE      *
0007A8 53780800              SUBQ   =1,DPC         ALERT DISPATCHER  *
0007AC 5368011E              SUBQ   =1,ACTC(A0)    FLAG 'NO TASK'    *
0007B0 08A800040120 IR6T05 BCLR    =4,ACSTR(A0)   ACIA FREE         *
0007B6 4A280119              TST.B  ACTOFF(A0)     TEST IF IR OFF NEED *
0007BA 6708                  BEQ.S  IR6T06         B:NO ACTION       *
0007BC 26680004              MOVEA.L RSR(A0),A3    GET STATUS ADR    *
0007C0 16A80119              MOVE.B ACTOFF(A0),(A3) TX-IR OFF        *
0007C4 4ED4            IR6T06 JMP    (A4)           IR-EXIT           *
0007C6                 *-----------------------------------------------*
0007C6                 *..... IR6-SUBROUTINE: UPDATE STATE OF CMMD ....*
0007C6 2268012A        IR6U00 MOVEA.L ACCTID(A0),A1 FETCH CMMD-TID    *
0007CA 1340003F              MOVE.B D0,BRKADR+1(A1) STORE TYPE OF CALL *
```

```
0007CE 4AA90016          TST.L  FDRS(A1)        IN IN DSP-CHAIN      *
0007D2 6620              BNE.S  IR6U02          B:IS LINKED          *
0007D4            *..... ACTIVATE DORM CMMD BY CYCBUF OF QERROR ...*
0007D4 34780830          MOVEA  EBWP,A2         WRITE-POINTER        *
0007D8 24C9              MOVE.L A1,(A2)+        STORE TID            *
0007DA 2278082A          MOVEA.L TIDQER,A1      TID OF QERROR        *
0007DE B4F80868          CMPA   EBU12+2,A2      MAX-TEST             *
0007E2 6004              BLT.S  IR6U01          B:NOT REACHED        *
0007E4 34780864          MOVEA  EBUCOM+2,A2     RESET POINTER        *
0007E8 B4F8082E   IR6U01 CMPA   EBRP,A2         IF READER REACHED    *
0007EC 6718              BEQ.S  IR6U03          B:DO NOT TOUCH PTR   *
0007EE 31CA0830          MOVE   A2,EBWP         NEW WRITER           *
0007F2 6012              BRA.S  IR6U03          GO, START QERROR     *
0007F4 4A29003F   IR6U02 TST.B  BRKADR+1(A1)    INSPECT TYPE OF CALL*
0007F8 660C              BNE.S  IR6U03          B:NOT RESTART        *
0007FA 4AA90026          TST.L  WTQ(A1)         IF THERE IS WSP      *
0007FE 6706              BEQ.S  IR6U03          B:NO RESTART         *
000800 08E900070044      BSET   =7,NESFLG(A1)   SET RESTART-FLAG     *
000806 022900CF0024 IR6U03 ANDI.B =#FF-BLKMSU-BLKMAC,BLOCK(A1)       *
00080C 53780800          SUBQ   =1,DPC          CALL DISPATCHER      *
000810 4ED4              JMP    (A4)            EXIT                 *
000812            *-----------------------------------------------------*
000812            *..... IR6-SUBROUTINE:   PUT DO INTO XMR-CYCBUF    *
000812 4A280121   IR6X00 TST.B  ACEMU(A0)       EMULATION ON ?       *
000816 661C              BNE.S  IR6X01          B:EMUL ON            *
000818 4A280118          TST.B  ACTON(A0)       TEST IF ACIA TYPE    *
00081C 6616              BNE.S  IR6X01          B:IN CASE OF ACIA    *
00081E 082800040120      BTST   =4,ACSTR(A0)    TEST IF READY TO WR  *
000824 660E              BNE.S  IR6X01          B: NOT READY         *
000826            *......ACIA-XMITTER IS READY TO SEND.............*
000826 26680000          MOVEA.L DAR(A0),A3     GET DATA REG-ADR     *
00082A 1680              MOVE.B D0,(A3)         WRITE                *
00082C 08E800040120      BSET   =4,ACSTR(A0)    FLAG 'NOT READY'     *
000832 4ED2              JMP    (A2)            RETURN FROM IR6X00   *
000834 3228010E   IR6X01 MOVE   ACTW(A0),D1     WRITE-POINTER        *
000838 D0FC0088          ADDA   =ACTB,A0        CYC.-BUFFER PTR      *
00083C 11801000          MOVE.B D0,0(A0,D1.W)   STORE THE BYTE       *
000840 90FC0088          SUBA   =ACTB,A0        OLD PTR              *
000844 5241              ADDQ   =1,D1           MOVE POINTER         *
000846 C27C007F          AND    =ACTBM,D1       MASK OFF             *
00084A B268010C          CMP    ACTR(A0),D1     IF READER REACHED    *
00084E 6704              BEQ.S  IR6X02          B:DO NOT STORE       *
000850 3141010E          MOVE   D1,ACTW(A0)     NEW WRITER-PTR       *
000854 4A280121   IR6X02 TST.B  ACEMU(A0)       IF EMUL. TERMINAL    *
000858 6714              BEQ.S  IR6X04          B:REAL TERMINAL      *
00085A C38A              EXG    A2,D1           SAVE RETURN          *
00085C 24680110          MOVEA.L ACRTID(A0),A2 RECEIVER-TASK-ID      *
000860 08AA00040024      BCLR   =BLKBSU,BLOCK(A2)  RUNNABLE          *
000866 53780800          SUBQ   =1,DPC          DISP. CALL           *
00086A C38A              EXG    A2,D1           RELOAD A2            *
00086C 4ED2       IR6X03 JMP    (A2)            START DISPATCHER     *
00086E 4A280118   IR6X04 TST.B  ACTON(A0)       TEST IF ACIA TYPE    *
000872 670C              BEQ.S  IR6X05          B:NO ACTION NEEDED   *
000874 220A              MOVE.L A2,D1           SAVE A2              *
000876 24680004          MOVEA.L RSR(A0),A2     GET COUPLER ADR      *
00087A 14A80118          MOVE.B ACTON(A0),(A2) TX IR ON              *
00087E 2441              MOVEA.L D1,A2          RELOAD A2            *
000880 4ED2       IR6X05 JMP    (A2)            EXIT                 *
000882            *-----------------------------------------------------*
000882            *..... IR6-SUBROUTINE:  PUT DO INTO INPUT-CE       *
000882            *      CARE: A4 IS DESTROYED!                       *
```

```
000882 082900010028 IR6Y00 BTST   =MODBBI,MODE(A1) TEST NO ECHO-MODE *
000888 664C                BNE.S  IR6Y06          BYPASS ALL ECHO ..  *
00088A                *..... TEST ON DELETE-CHAR                     *
00088A 284A                MOVEA.L A2,A4          SAVE RETURN ADR     *
00088C B03C007F            CMP.B  =$7F,D0         TEST ON DELETE      *
000890 6706                BEQ.S  IR6Y01          B:IS A DELETE       *
000892 B03C0008            CMP.B  =$08,D0         TEST ON DELETE      *
000896 6634                BNE.S  IR6Y04          B:NORMAL ECHO       *
000898                *.... REPOS BUFFER-POINTER  OR NO ACTION .......*
000898 2468012E    IR6Y01  MOVEA.L ACRCAD(A0),A2  READER ADR IN CE    *
00089C 538A                SUBQ.L =1,A2           RESET.              *
00089E B5E90020            CMPA.L BUADR(A1),A2    TEST UNDERFLOW      *
0008A2 6C02                BGE.S  IR6Y02          B:NO UNDERFLOW      *
0008A4 4ED4                JMP    (A4)            RETURN - NO ACTION  *
0008A6                *..... REPOSITION BY ONE BYTE  ................ *
0008A6 214A012E    IR6Y02  MOVE.L A2,ACRCAD(A0)   STORE POINTER       *
0008AA 53690024            SUBQ   =1,RECLEN(A1)   REDUCE BY ONE       *
0008AE 5268011C            ADDQ   =1,ACRC(A0)     BYTES TO READ       *
0008B2 7008                MOVEQ  =$08,D0         CURS. LEFT          *
0008B4 45FA0006            LEA    IR6Y03,A2       RETURN-ADR          *
0008B8 6000FF58            BRA    IR6X00          XMIT DO (D0=$08)    *
0008BC 7020        IR6Y03  MOVEQ  =$20,D0         ONE BLANK           *
0008BE 508A                ADDQ.L =8,A2           NEW RET ADR         *
0008C0 6000FF50            BRA.L  IR6X00          XMIT THE CHAR       *
0008C4 7008                MOVEQ  =$08,D0         <-                  *
0008C6 244C                MOVEA.L A4,A2          EXIT = RETURN       *
0008C8 6000FF48            BRA    IR6X00          WRITE THE CHAR+EXIT *
0008CC                *..... ECHO NORMAL CHARACTER ...................*
0008CC 45FA0006    IR6Y04  LEA    IR6Y05,A2       RETURN-ADR          *
0008D0 6000FF40            BRA    IR6X00          XMIT DO             *
0008D4 244C        IR6Y05  MOVEA.L A4,A2          RELOAD RETURN       *
0008D6 2868012E    IR6Y06  MOVEA.L ACRCAD(A0),A4  READER-ADR IN CE    *
0008DA 18D0                MOVE.B D0,(A4)+        STORE CHARACTER     *
0008DC 214C012E            MOVE.L A4,ACRCAD(A0)   NEW BYTE-ADR        *
0008E0 52690024            ADDQ   =1,RECLEN(A1)   INCREMENT           *
0008E4 5368011C            SUBQ   =1,ACRC(A0)     READER-COUNTER      *
0008E8 6F02                BLE.S  IR6Y09          B:WAS LAST CHARACTER*
0008EA 7220                MOVEQ  =MODMCR,D1      CR-MASK             *
0008EC 903C000D            SUB.B  =$0D,D0         TEST ON CR          *
0008F0 6E40                BGT.S  IR6Y0A          B:NO SPECIAL CHAR   *
0008F2 6722                BEQ.S  IR6Y08          B:YES IS A CR       *
0008F4 7210                MOVEQ  =MODMLF,D1      MASK LINE-FEED      *
0008F6 5600                ADDQ.B =$0D-$0A,D0     TEST ON LF          *
0008F8 6616                BNE.S  IR6Y07          B:YES IS NOT LF     *
0008FA 082900000002B       BTST   =0,DRIVE+1(A1)    TEST 'UL'-OPTION  *
000900 6714                BEQ.S  IR6Y08          B:NO SPECIAL ACTION *
000902                *..... LINE-FEED AND 'UL' IN NOT PURE RECLEN-MODE*
000902 5268011C            ADDQ   =1,ACRC(A0)     READJUST READER-CNT *
000906 53690024            SUBQ   =1,RECLEN(A1)       ''              *
00090A 53A8012E            SUBQ.L =1,ACRCAD(A0)   BUFFER-POINTER      *
00090E 4ED2                JMP    (A2)            IGNORE LF TOTALLY   *
000910 7208        IR6Y07  MOVEQ  =MODMEO,D1      MASK FOR EOT        *
000912 5C00                ADDQ.B =$0A-$04,D0     TEST ON EOT         *
000914 661C                BNE.S  IR6Y0A          B:NOT CR,LF,EOT     *
000916 C2290028    IR6Y08  AND.B  MODE(A1),D1     TEST IF MATCHING    *
00091A 6716                BEQ.S  IR6Y0A          B:NO MATCH          *
00091C                *..... STOP READING: CE FULL OR CR/LF/EOT BY MODE*
00091C 4268011C    IR6Y09  CLR    ACRC(A0)        CLEAR READER-COUNT  *
000920 4228011A            CLR.B  ACIS(A0)        INPUT-SWITCH        *
000924 28680110            MOVEA.L ACRTID(A0),A4  TID OF TRANSMITTER  *
000928 0BAC00040024        BCLR   =BLKBSU,BLOCK(A4)  CONTINUE         *
```

```
00092E 53780800              SUBQ    =1,DPC          CALL DISPATCHER     *
000932 4ED2          IR6Y0A  JMP     (A2)            RETURN              *
000934                       *-----------------------------------------*
000934                       *..... MALFUNCTION-ENTRY + INTERRUPT-EXIT .....*
000934 0839000400FF
       FA01          IR6Z00  BTST    =4,MGPIP        MORE IR ?           *
00093C 6700FBEA              BEQ     IR611X          B: YES              *
000940 4CDF1F03      IR6Z01  MOVEM.L (A7)+,D0/D1/A0/A1/A2/A3/A4 RELOAD   *
000944 31DF07FE              MOVE    (A7)+,IID       RELOAD IID          *
000948                       *..... DISPATCHER-EXIT THIS SLICE ..............*
000948 007C0700      DISP    ORI     =$700,SR        STOP ALL IR         *
00094C 4A780800              TST     DPC             IF A CALL           *
000950 6802                  BMI.S   DISP2           B:YES               *
000952 4E73                  RTE                                         *
000954 4EF80906      DISP2   JMP     DISEX+JUMTBL    START DEFINITELY    *
000958                       *-----------------------------------------*
000958                       *..... CLOCK-SECTION ...........................*
000958 0028                  DC      IR6MF-IR6114    MALFUNCTION-ADR     *
00095A 5388094C      IR6114  SUBQ.L =1,TIMEC         TIME CORRECTION     *
00095E 6E0C                  BGT.S   IR611A          B: NO CORRECTION    *
000960 21FC00000060
       094C                  MOVE.L =96,TIMEC        FOR NEXT ROUND      *
000968 528808BA              ADDQ.L =1,TIME          CORRECTION          *
00096C 5888088A      IR611A  ADDQ.L =CLTF,TIME       TIME INCREMENT      *
000970 6A02                  BPL.S   IR611B          B:MARKED TIME       *
000972 4E73                  RTE                                         *
000974 3F3807FE      IR611B  MOVE    IID,-(A7)       SAVE IID            *
000978 31FC011407FE          MOVE    =IRCLIV,IID     LOAD NEW IID        *
00097E 4EF80900              JMP     CLKIR+JUMTBL    PROCESS MARKED TIME *
000982                       *..... PRESCRIBED MALFUNCTION PROCESS ..........*
000982 4CDF02E2      IR6MF   MOVEM.L (A7)+,D1/D5-D7/A1  FROM NUKLEUS     *
000986 31DF07FE              MOVE    (A7)+,IID       RELOAD IID          *
00098A 6000FFBC              BRA     DISP            EXIT                *
00098E                       *                                           *
00098E                       ***************************************************
00098E                       *                                           *
00098E                       *..... '#ACIA1'-DRIVER-TASK ....................*
00098E 4E49                  QRELCE                  FOR RETURN-JUMP     *
000990 7200          QACIA1  MOVEQ   =0,D1           OWN LDN             *
000992 48FAFFFA              LEA     QACIA1-2,A5     RETURN-ADR          *
000996 20780846              MOVEA.L IDP6,A0         BUFFER-INDEX        *
00099A 117C00010121          MOVE.B  =1,ACEMU(A0)    SET EMULATION ON    *
0009A0 604A                  BRA.S   QACIAX          TO ACIA-COMMON      *
0009A2                       *-----------------------------------------*
0009A2                       *..... '#RS232'-DRIVER-TASK ....................*
0009A2 4E49                  QRELCE                                      *
0009A4 20780846      QACIA2  MOVEA.L IDP6,A0         BUFFER-INDEX        *
0009A8 41E80136              LEA     ACLEN(A0),A0    USART BUFFER-INDEX  *
0009AC 7202                  MOVEQ   =2,D1           OWN LDN             *
0009AE 48FAFFF2              LEA     QACIA2-2,A5     RETURN-ADR          *
0009B2 42280121              CLR.B   ACEMU(A0)       NOT EMUL.           *
0009B6 2278084E              MOVEA.L SIODDT,A1       I/O-DEV TABLE       *
0009BA 082900020005          BTST    =2,5(A1)        TEST EMUL-BIT LDN 2 *
0009C0 672A                  BEQ.S   QACIAX          B:REAL TERMINAL     *
0009C2 52280121              ADDQ.B  =1,ACEMU(A0)    FLAG EMUL. TERMINAL *
0009C6 6024                  BRA.S   QACIAX          TO ACIA-COMMON      *
0009C8                       *-----------------------------------------*
0009C8                       *..... '#MIDI3'-DRIVER-TASK ....................*
0009C8 4E49                  QRELCE                  FOR RETURN-JUMP     *
0009CA 20780846      QACIA3  MOVEA.L IDP6,A0         BUFFER-INDEX        *
0009CE 41E8026C              LEA  2*ACLEN(A0),A0     USART BUFFER-INDEX  *
```

```
0009D2 7204                  MOVEQ  =4,D1           OWN LDN                 *
0009D4 4BFAFFF2              LEA    QACIA3-2,A5     RETURN-ADR              *
0009D8 42280121              CLR.B  ACEMU(A0)       NOT EMUL.               *
0009DC 2278084E              MOVEA.L SIODDT,A1      I/O-DEV TABLE           *
0009E0 082900020005          BTST   =2,5(A1)        TEST EMUL-BIT LDN 2     *
0009E6 6704                  BEQ.S  QACIAX          B:REAL TERMINAL         *
0009E8 52280121              ADDQ.B =1,ACEMU(A0)    FLAG EMUL. TERMINAL     *
0009EC                 *-----------------------------------------------------*
0009EC                 *..... REENTRANT SUBROUTINE FOR QACIA 1/2/3 .....*
0009EC 2C780802        QACIAX MOVEA.L TID,A6         LOAD OWN TID            *
0009F0 286E0026              MOVEA.L WTO(A6),A4     LOAD PTN TO TWSP        *
0009F4 4E4D                  QTOQ                   LOAD A1 FROM CE         *
0009F6 4E41                  QTERMI                 IF EMPTY                *
0009F8 4FEC00F2        QACIX0 LEA    OPNAME+140.T,A7 LOAD STACK PTR         *
0009FC 022800770120          ANDI.B =$77,ACSTR(A0) ENABLE COMMANDS          *
000A02 082900000028          BTST   =MODBPT,MODE(A1) BINARY MODE            *
000A08 6706                  BEQ.S  QACXX0          B: NO BINARY MODE       *
000A0A 08E800030120          BSET   =3,ACSTR(A0)    SET TO BIN-MODE         *
000A10 082900020028    QACXX0 BTST   =MODBSC,MODE(A1) COMMANDS SUPRESS?     *
000A16 6706                  BEQ.S  QACIX1          B: COMM. ENABLED        *
000A18 08E800070120          BSET   =7,ACSTR(A0)    (ACSTR) NO COMMANDS     *
000A1E 082900060028    QACIX1 BTST   =MODBOU,MODE(A1)  IN OR OUT?           *
000A24 6600018A              BNE    QACIX0          B:GO OUTPUT-SECTION     *
000A28                 *..... INPUT-SECTION ON GROUND-LEVEL ............*
000A28 214E0110              MOVE.L A6,ACRTID(A0)   RECEIVER-TID            *
000A2C 21490122              MOVE.L A1,ACRA1(A0)    RECEIVER-A1             *
000A30 21690020012E          MOVE.L BUADR(A1),ACRCAD(A0)  START-ADR         *
000A36 7E06                  MOVEQ  =6,D7           MASK B OR C             *
000A38 34290024              MOVE   RECLEN(A1),D2   ACCESS                  *
000A3C 6602                  BNE.S  $+4             B:NOT EMPTY CE          *
000A3E 4ED5                  JMP    (A5)            EXIT, NO ACTION         *
000A40 4E4F                  IROFF                  STOP IR                 *
000A42 36280108              MOVE   ACRR(A0),D3     READ-POINTER            *
000A46 2878083E              MOVEA.L IDP4,A4        BUFFER-PTR              *
000A4A 08AC00030002          BCLR   =KEYF,STAT1(A4) FUNK.-KEY PRESSED?     *
000A50 6706                  BEQ.S  QACI11          B: NO NORMAL            *
000A52 286E0026              MOVEA.L WTO(A6),A4     RESTORE OLD PTR         *
000A56 602E                  BRA.S  QACIX3          B: 'B'-PORT             *
000A58 286E0026        QACI11 MOVEA.L WTO(A6),A4     RESTORE OLD PTR         *
000A5C CE29002B              AND.B  DRIVE+1(A1),D7  TEST NOR B OR C         *
000A60 67000088              BEQ    QACIX9          B:GO DIRECT READ        *
000A64 082900020028          BTST   =2,DRIVE+1(A1)  TEST ON 'C'-OPTION      *
000A6A 671A                  BEQ.S  QACIX3          B:IS 'B'-OPTION         *
000A6C                 *..... 'C'-TYPE PORT-XFER ......................*
000A6C 3428010A              MOVE   ACRW(A0),D2     WRITE-POINTER           *
000A70 9443                  SUB    D3,D2           COMPUTE LENGTH OF       *
000A72 6E08                  BGT.S  QACIX2          B:LENGTH IN D2          *
000A74 67000128              BEQ    QACIXN          B:TRANSMIT A NUL        *
000A78 D47C0080              ADD    =ACRBM+1,D2     PRECISE LENGTH          *
000A7C B4690024        QACIX2 CMP RECLEN(A1),D2      COMPARE TO CE           *
000A80 6F04                  BLE.S  QACIX3          B:TAKE WHAT IS          *
000A82 34290024              MOVE   RECLEN(A1),D2   LOAD RECLEN AGAIN       *
000A86                 *..... 'B' AND 'C' OPTION PROCESSING ............*
000A86 3142011C        QACIX3 MOVE   D2,ACRC(A0)     INITIALIZE READ-COUN*
000A8A 08E800010120          BSET   =1,ACSTR(A0)    FLAG B-OPTION           *
000A90 42690024              CLR    RECLEN(A1)      NO BYTE XFERRRED        *
000A94 007C2700        QACIX4 ORI    =$2700,SR       ALL IR 'OFF'            *
000A98 34280108              MOVE   ACRR(A0),D2     READER                  *
000A9C B468010A              CMP    ACRW(A0),D2     TEST EMPTY              *
000AA0 675A                  BEQ.S  QACIXA          B:WAIT FOR DATA         *
000AA2 10302008              MOVE.B ACRB(A0,D2.W),D0  GET THE CHARACTER*
```

```
000AA6 5242                 ADDQ   =1,D2           MOVE-UP                 *
000AA8 C47C007F             AND    =ACRBM,D2       MODULO-OP               *
000AAC 31420108             MOVE   D2,ACRR(A0)     STORE NEW POINTER       *
000AB0 45FA0006             LEA    QACIX5,A2       RETURN-ADR              *
000AB4 6000FDCC             BRA    IR6Y00          STORE ONE BYTE          *
000AB8 46FC2300     QACIX5  MOVE   =$2300,SR       ENABLE INTERRUPTS       *
000ABC 4A68011C             TST    ACRC(A0)        IF ALL DONE?            *
000AC0 6ED2                 BGT.S  QACIX4          LOOPING                 *
000AC2              *.....  CE WAS SATISFIED FROM BUFFER   ...........*
000AC2 007C2700             ORI    =$2700,SR       ALL IR 'OFF'            *
000AC6 34280108             MOVE   ACRR(A0),D2     READER                  *
000ACA B468010A             CMP    ACRW(A0),D2     TEST EMPTY              *
000ACE 6602                 BNE.S  QACIX6          B: NOT EMPTY            *
000AD0              *......CYCLIC BUFFER IS EMPTY SEND X-ON..........*
000AD0 6108                 BSR.S  QACIX8          SEND 'X-ON' ON COND.*
000AD2 027CD8FF     QACIX6  ANDI   =$D8FF,SR       SWITCH IR ON            *
000AD6 615A         QACIX7  BSR.S  QACIXE          EMIT XMR-BUFFER I/A *
000AD8 4ED5                 JMP    (A5)            RETURN TO CALLER        *
000ADA              *....   SEND X-ON ON CONDITION ................... *
000ADA 245F         QACIX8  MOVEA.L (A7)+,A2       RETURN-ADR              *
000ADC 7011                 MOVEQ  =$11,D0         LOAD CHAR               *
000ADE 08A800020120         BCLR   =2,ACSTR(A0)    TEST AND RESET          *
000AE4 6600FD2C             BNE    IR6X00          B:SEND 'X-ON'           *
000AE8 4ED2                 JMP    (A2)            EXIT, NO ACTION         *
000AEA              *.....  ENTRY FOR UNBUFFERED READ ................*
000AEA 3143010A     QACIX9  MOVE   D3,ACRW(A0)     CLEAR BUFFER W=R        *
000AEE 3142011C             MOVE   D2,ACRC(A0)     READER-COUNT            *
000AF2 08A800010120         BCLR   =1,ACSTR(A0)    FLAG 'NO PROTOCOL'      *
000AF8 42690024             CLR    RECLEN(A1)      RESET COUNTER           *
000AFC              *.....  CYCLIC INPUT-BUFFER IS EMPTY, SEND X-ON       *
000AFC              *.....  A,B,C  COMMON INPUT CODE-SECTION               *
000AFC 61DC         QACIXA  BSR.S  QACIX8          SEND X-ON ON COND.      *
000AFE 4A280121             TST.B  ACEMU(A0)       IF EMULATED TERMINAL*
000B02 670A                 BEQ.S  QACIXB          B:NOT EMUL. TERMINAL*
000B04 027CD8FF             ANDI   =$D8FF,SR       IR ON                   *
000B08 61000030             BSR    QACIXF          MAKE THE ECHO           *
000B0C 4E4F                 IROFF                  STOP IR                 *
000B0E 5228011A     QACIXB  ADDQ.B =1,ACIS(A0)     SWITCH INPT TO CE       *
000B12 08EE00040024 QACIXC  BSET   =BLKBSU,BLOCK(A6)  STOP THE TASK       *
000B18 027CD8FF             ANDI   =$D8FF,SR       IR ON                   *
000B1C 4E43                 QDPC                   MAKE THE STOP           *
000B1E 4A280121             TST.B  ACEMU(A0)       IF EMULATED TERMINAL*
000B22 67B2                 BEQ.S  QACIX7          B:NOT EMUL. TERMINAL*
000B24 61000014             BSR    QACIXF          MAKE THE ECHO           *
000B28 4E4F                 IROFF                  STOP IR                 *
000B2A 4A28011A             TST.B  ACIS(A0)        IF STILL ON CE          *
000B2E 66E2                 BNE.S  QACIXC          LOOPING, WAIT NEXT      *
000B30 60A0                 BRA.S  QACIX6          LAST ECHO THEN EXIT     *
000B32              *.....  SUBROUT: ECHO ALL CHAR FROM XMR-CYC-BUF       *
000B32 4A280121     QACIXE  TST.B  ACEMU(A0)       IF EMUL ON              *
000B36 6602                 BNE.S  QACIXF          B:YES EMUL ON           *
000B38 4E75                 RTS                                            *
000B3A 3228010C     QACIXF  MOVE   ACTR(A0),D1     READ-POINTER            *
000B3E B268010E             CMP    ACTW(A0),D1     INSPECT FOR DATA        *
000B42 6602                 BNE.S  $+4             B:NOT EMPTY, SKIP       *
000B44 4E75                 RTS                    EXIT, ALL DONE          *
000B46 D0FC0088             ADDA   =ACTB,A0        LOAD PTR                *
000B4A 10301000             MOVE.B 0(A0,D1.W),D0   GET THE BYTE            *
000B4E 90FC0088             SUBA   =ACTB,A0        OLD PTR                 *
000B52 5241                 ADDQ   =1,D1           MOVE POINTER            *
000B54 C27C007F             AND    =ACTBM,D1       MASK                    *
```

```
000B58 3141010C                MOVE    D1,ACTR(A0)     NEW POINTER          *
000B5C 48E760F0                MOVEM.L D1/D2/A0/A1/A2/A3,-(A7)  SAVE        *
000B60 2678083E                MOVEA.L IDP4,A3         INDEX-BUFFER         *
000B64 08EB00070002            BSET    =ACTION,STAT1(A3) CHAR IN SERVICE    *
000B6A 082B00060002            BTST    =CURSF,STAT1(A3) CURSOR ON/OFF ?     *
000B70 6708                    BEQ.S   QACIXG          B: IF OFF            *
000B72 20780836                MOVEA.L IDP2,A0         GET JUMPTABEL        *
000B76 4EA80004                JSR     CHAINV(A0)      MAKE CURSOR OFF      *
000B7A 20780836        QACIXG  MOVEA.L IDP2,A0         GET JUMPTABEL        *
000B7E 4EA80008                JSR     SCREEN(A0)      CALL EMULATOR        *
000B82 082B00060002            BTST    =CURSF,STAT1(A3) CURSOR ON/OFF       *
000B88 6708                    BEQ.S   QACIXH          B: IF OFF            *
000B8A 20780836                MOVEA.L IDP2,A0         GET JUMPTABEL        *
000B8E 4EA80004                JSR     CHAINV(A0)      MAKE CURSOR ON       *
000B92 08AB00070002    QACIXH  BCLR    =ACTION,STAT1(A3) CHAR  OK           *
000B98 4CDF0F06                MOVEM.L (A7)+,D1/D2/A0/A1/A2/A3      RELOAD*
000B9C 609C                    BRA.S   QACIXF          LOOP                 *
000B9E                 *..... EMPTY-BUFFER IN 'C'-OPTION ...............*
000B9E 027CD8FF        QACIXN  ANDI    =$D8FF,SR       IR-SYSTEM 'ON'       *
000BA2 24690020                MOVEA.L BUADR(A1),A2    TEXT ADR             *
000BA6 4212                    CLR.B   (A2)            A NUL                *
000BA8 337C00010024            MOVE    =1,RECLEN(A1)   ONE CHAR.            *
000BAE 4ED5                    JMP     (A5)            RETURN TO CALLER     *
000BB0                 *..... OUTPUT-SECTION  ACIA ....................*
000BB0 214E0114        QACIXO  MOVE.L A6,ACTTID(A0)    TRANSM. TID          *
000BB4 34290024                MOVE    RECLEN(A1),D2   ASSUMED LENGTH       *
000BB8 7C38                    MOVEQ   =MODMCR+MODMLF+MODMEO,D6             *
000BBA CC290028                AND.B   MODE(A1),D6     QUICK CHECK          *
000BBE 672A                    BEQ.S   QACIXR          B:NO DATA SCAN       *
000BC0 24690020                MOVEA.L BUADR(A1),A2    TEXT-ORG             *
000BC4 4243                    CLR     D3              RESET RECLEN         *
000BC6 B642            QACIXP  CMP     D2,D3           RECLEN-TEST          *
000BC8 6C20                    BGE.S   QACIXR          B:ALL DONE           *
000BCA 5243                    ADDQ    =1,D3           COUNTER              *
000BCC 101A                    MOVE.B  (A2)+,D0        INSPECT THE BYTE     *
000BCE 7E20                    MOVEQ   =MODMCR,D7      TEST-FRAME           *
000BD0 903C000D                SUB.B   =$0D,D0         TEST FOR CR          *
000BD4 6EF0                    BGT.S   QACIXP          B:TAKE NEXT          *
000BD6 670C                    BEQ.S   QACIXQ          B:GO FOR CHECK       *
000BD8 7E10                    MOVEQ   =MODMLF,D7      TEST-MASK            *
000BDA 5600                    ADDQ.B  =$0D-$0A,D0     TEST FOR LF          *
000BDC 6706                    BEQ.S   QACIXQ          B:GO FOR TEST        *
000BDE 5C00                    ADDQ.B  =$0A-$04,D0     TEST FOR EOT         *
000BE0 66E4                    BNE.S   QACIXP          TEST NEXT CHAR.      *
000BE2 7E08                    MOVEQ   =MODMEO,D7      TEST MASK EOT        *
000BE4 CE06            QACIXQ  AND.B   D6,D7           MODE-MATCH?          *
000BE6 67DE                    BEQ.S   QACIXP          B:NO MATCH           *
000BE8 3403                    MOVE    D3,D2           RECLEN-UPDATE        *
000BEA 24690020        QACIXR  MOVEA.L BUADR(A1),A2    FOR EMUL. TERM.      *
000BEE 214A0132                MOVE.L A2,ACTCAD(A0)    TEXT-ADR             *
000BF2 4A280121                TST.B   ACEMU(A0)       IF EMUL TERM.        *
000BF6 6634                    BNE.S   QACIXW          B:EMUL./ TERMINAL    *
000BF8 4E4F                    IROFF                   STOP   ALL IR        *
000BFA 3142011E                MOVE    D2,ACTC(A0)     COUNTER-INIT         *
000BFE 6724                    BEQ.S   QACIXS          B:NO JOB             *
000C00 08EE00040024            BSET    =BLKBSU,BLOCK(A6)  SUSPEND           *
000C06 4A280118                TST.B   ACTON(A0)       TEST ACIA TYPE       *
000C0A 6610                    BNE.S   QACIXT          B:ACIA TYPE          *
000C0C 082800040120            BTST    =4,ACSTR(A0)    READY TO SEND ?      *
000C12 6610                    BNE.S   QACIXS          B: NOT READY         *
000C14 49FA000E                LEA     QACIXS,A4       RETURN ADR           *
```

```
000C18 6000F82E            BRA     IR&TOO          GO TRANSMITTER SECT *
000C1C                 *......ACIA IR ON ...............................*
000C1C 24680004        QACIXT MOVEA.L RSR(A0),A2      GET COUPLER ADR     *
000C20 14A80118            MOVE.B  ACTON(A0),(A2) SWITCH ACIA IR ON   *
000C24 027CD8FF        QACIXS AND1    =$D8FF,SR       IR-ON               *
000C29 4E43                QDPC                       STOP THE TASK       *
000C2A 4ED5                JMP     (A5)               EXIT                *
000C2C                 *..... OUTPUT TO EMUL. TERMINAL .................*
000C2C 48E76090        QACIXW MOVEM.L D1/D2/A0/A3,-(A7) SAVE REG.         *
000C30 2678083E            MOVEA.L IDP4,A3            BUFFER-POINTER      *
000C34 08E800070002        BSET    =ACTION,STAT1(A3) CHAR IN SERVICE   *
000C3A 082800060002        BTST    =CURSF,STAT1(A3) CURSOR ON/OFF      *
000C40 6708                BEQ.S   QACIW1             B: IF OFF           *
000C42 20780836            MOVEA.L IDP2,A0            GET JUMPTABEL       *
000C46 4EA80004            JSR     CHAINV(A0)         MAKE CURSOR OFF     *
000C4A 4CDF0906        QACIW1 MOVEM.L (A7)+,D1/D2/A0/A3 RELOAD            *
000C4E 5342            QACIW2 SUBQ    =1,D2           COUNTER             *
000C50 6C24                BGE.S   QACIXY             B:THERE IS DATA     *
000C52 48E76090            MOVEM.L D1/D2/A0/A3,-(A7) SAVE REG.         *
000C56 2678083E            MOVEA.L IDP4,A3            BUFFER-PTR.         *
000C5A 082800060002        BTST    =CURSF,STAT1(A3) CURSOR ON ?        *
000C60 6708                BEQ.S   QACIW3             B: IS OFF           *
000C62 20780836            MOVEA.L IDP2,A0            GET JUMPTABEL       *
000C66 4EA80004            JSR     CHAINV(A0)         MAKE CURSOR ON      *
000C6A 08A800070002 QACIW3 BCLR    =ACTION,STAT1(A3) CHAR OK           *
000C70 4CDF0906            MOVEM.L (A7)+,D1/D2/A0/A3 RELOAD           *
000C74 4ED5                JMP     (A5)               ALL DONE, RETURN    *
000C76 082800000120 QACIXY BTST    =0,ACSTR(A0)       TEST IF STOPPED     *
000C7C 6710                BEQ.S   QACIXZ             GO AND SEND ONE BYTE*
000C7E 297C80000008
       006C                MOVE.L  =$80000008,OPFATI.T  RESUME-TIME   *
000C86 2F09                MOVE.L A1,-(A7)            SAVE A1             *
000C88 A02C                QTIRE                      TIME-RESUME         *
000C8A 225F                MOVEA.L (A7)+,A1           RELOAD A1           *
000C8C 60E8                BRA.S   QACIXY             REPEAT THE TEST     *
000C8E 101A            QACIXZ MOVE.B (A2)+,D0         LOAD THE CHARACTER  *
000C90 48E76F0             MOVEM.L D1/D2/A0/A1/A2/A3,-(A7)     SAVE   *
000C94 2678083E            MOVEA.L IDP4,A3            BUFFER-INDEX        *
000C98 20780836            MOVEA.L IDP2,A0            GET JUMPTABEL       *
000C9C 4EA80008            JSR     SCREEN(A0)         EMUL. TERMINAL      *
000CA0 4CDF0F06            MOVEM.L (A7)+,D1/D2/A0/A1/A2/A3      RELOAD*
000CA4 60A8                BRA.S   QACIW2                                 *
000CA6                 *-----------------------------------------------*
000CA6                 *......SEND A BYTE FROM D0.B ....................*
000CA6                 *      A0  POINTER TO IRSBUFF                     *
000CA6                 *      A3  RETURN ADR                             *
000CA6 13FC000F00FF
       8800            PPOUT  MOVE.B =$0F,PSGA           SELECT PORT B   *
000CAE 13C000FF8802            MOVE.B D0,PSGD             OUTPUT DATA     *
000CB4 13FC000E00FF
       8800                   MOVE.B =$0E,PSGA           SELECT PORT A   *
000CBC 022800DF03A2            ANDI.B =$DF,STAPSG(A0) PREPEARE STROBE LOW*
000CC2 13E803A200FF
       8802                   MOVE.B STAPSG(A0),PSGD SET STROBE LOW      *
000CCA 002800200 3A2           ORI.B  =$20,STAPSG(A0) PREPEARE STROBE HIG*
000CD0 13E803A200FF
       8802                   MOVE.B STAPSG(A0),PSGD SET STROBE HIGH     *
000CD8 4ED3                   JMP     (A3)               RETURN TO CALLER *
000CDA                 *......CENTRONIX DRIVER TASK ....................*
000CDA 4E49            QCENTO QRELCE                     RELEASE THE CE  *
000CDC 720A            QCENT  MOVEQ  =PPLDN,D1           GET OWN LDN     *
```

```
000CDE 20780846                MOVEA.L IDP6,A0          GET POINTER TO IR BU*
000CE2 2C780802                MOVEA.L TID,A6           GET TID OF OWN TASK *
000CE6 4E4D                    QTOQ                     TAKE OF QUEUE       *
000CE8 4E41                    QTERMI                   IF NO CE THERE      *
000CEA 214E03A4                MOVE.L A6,PPTID(A0)      TRANSM. TID         *
000CEE 34290024                MOVE   RECLEN(A1),D2     ASSUMED LENGTH      *
000CF2 7C38                    MOVEQ  =MODMCR+MODMLF+MODMEO,D6              *
000CF4 CC290028                AND.B  MODE(A1),D6       QUICK CHECK         *
000CF8 672A                    BEQ.S  QCENTR            B:NO DATA SCAN      *
000CFA 24690020                MOVEA.L BUADR(A1),A2     TEXT-ORG            *
000CFE 4243                    CLR    D3                RESET RECLEN        *
000D00 B642            QCENTP  CMP    D2,D3             RECLEN-TEST         *
000D02 6C20                    BGE.S  QCENTR            B:ALL DONE          *
000D04 5243                    ADDQ   =1,D3             COUNTER             *
000D06 101A                    MOVE.B (A2)+,D0          INSPECT THE BYTE    *
000D08 7E20                    MOVEQ  =MODMCR,D7        TEST-FRAME          *
000D0A 903C000D                SUB.B  =$0D,D0           TEST FOR CR         *
000D0E 6EF0                    BGT.S  QCENTP            B:TAKE NEXT         *
000D10 670C                    BEQ.S  QCENTQ            B:GO FOR CHECK      *
000D12 7E10                    MOVEQ  =MODMLF,D7        TEST-MASK           *
000D14 5600                    ADDQ.B =$0D-$0A,D0       TEST FOR LF         *
000D16 6706                    BEQ.S  QCENTQ            B:GO FOR TEST       *
000D18 5C00                    ADDQ.B =$0A-$04,D0       TEST FOR EOT        *
000D1A 66E4                    BNE.S  QCENTP            TEST NEXT CHAR.     *
000D1C 7E08                    MOVEQ  =MODMEO,D7        TEST MASK EOT       *
000D1E CE06            QCENTQ  AND.B  D6,D7             MODE-MATCH?         *
000D20 67DE                    BEQ.S  QCENTP            B:NO MATCH          *
000D22 3403                    MOVE   D3,D2             RECLEN-UPDATE       *
000D24 24690020        QCENTR  MOVEA.L BUADR(A1),A2     GET TEXT START ADR  *
000D28 214A03AA                MOVE.L A2,PPCAD(A0)      TEXT-ADR            *
000D2C 4E4F                    IROFF                    STOP   ALL IR       *
000D2E 314203AE                MOVE   D2,PPCNT(A0)      COUNTER-INIT        *
000D32 6728                    BEQ.S  QCENTT            B:NO JOB            *
000D34 0828000003A8            BTST   =0,PPSTAT(A0)     IS THE PORT FREE ?  *
000D3A 671A                    BEQ.S  QCENTU            B:IF NOT            *
000D3C 101A                    MOVE.B (A2)+,D0          GET DATA            *
000D3E 47FA0006                LEA    QCENTS,A3         RETURN ADR          *
000D42 6000FF62                BRA    PPOUT             SEND THE BYTE       *
000D46 08A8000003A8    QCENTS  BCLR    =0,PPSTAT(A0)    PORT NOT FREE       *
000D4C 214A03AA                MOVE.L A2,PPCAD(A0)      STORE DATA ADR      *
000D50 536803AE                SUBQ   =1,PPCNT(A0)      DECREMENT BYTE COUNT*
000D54 6706                    BEQ.S  QCENTT            IF JOB IS DONE      *
000D56 08EE00040024    QCENTU  BSET   =BLKBSU,BLOCK(A6)  SUSPEND            *
000D5C 027CDBFF        QCENTT  ANDI   =$DBFF,SR         IR-ON               *
000D60 4E43                    QDPC                     STOP THE TASK       *
000D62 6000FF76                BRA    QCENTO            EXIT                *
000D66                 *                                                     *
000D66                 ***********************************************************
000D66                 *                                                     *
000D66                 *......TER.EMUL.-PARAMETER INSIDE BUFFER.........*
000D66                 *                                                     *
000D66 00000000        TERMD   EQU    0                 ADR-LINK TERM       *
000D66 00000002        STAT1   EQU    TERMD+2           TASTATUR STATUS     *
000D66 00000004        STAT2   EQU    STAT1+2           HIGH/LOW RES. STATUS*
000D66 00000006        STAT3   EQU    STAT2+2           ESC STATUS          *
000D66 00000008        COLOUR  EQU    STAT3+2           AKTUEL COLOUR       *
000D66 0000000A        LINE    EQU    COLOUR+2          AKTUEL LINENUMBER   *
000D66 0000000C        COLUMN  EQU    LINE+2            AKTUEL COLUMNNUMBER *
000D66 0000000E        CURT    EQU    COLUMN+2          CURSOR BLINK DELAY  *
000D66 00000010        CHAR    EQU    CURT+2            CHAR. FOR AUTO REP. *
000D66 00000012        AUTOD   EQU    CHAR+2            DELAY RATE COUNTER  *
```

```
000D66 00000014    AUTOS  EQU    AUTOD+2        STEP RATE COUNTER    *
000D66 00000016    KEYADR EQU    AUTOS+2        MAUS RECEIVER PTR    *
000D66 00000020    SCRLEN EQU    KEYADR+10      SCREEN LENGTH 24/25  *
000D66 00000022    SCRCO1 EQU    SCRLEN+2       COUNT FOR CLR/SCROLL*
000D66 00000024    SCRCO2 EQU    SCRCO1+2       COUNT FOR SMOTH      *
000D66 00000026    RET    EQU    SCRCO2+2       PLACE FOR SAVE       *
000D66 0000002A    TEMP   EQU    RET+4
000D66 0000002C    I4LEN  EQU    TEMP+2         FUNKTION BUFFER      *
000D66             *-----------------------------------------------*
000D66 00000000    F1     EQU    0                                   *
000D66 00000050    F2     EQU    F1+80                               *
000D66 000000A0    F3     EQU    F2+80                               *
000D66 000000F0    F4     EQU    F3+80                               *
000D66 00000140    F5     EQU    F4+80                               *
000D66 00000190    F6     EQU    F5+80                               *
000D66 000001E0    F7     EQU    F6+80                               *
000D66 00000230    F8     EQU    F7+80                               *
000D66 00000280    F9     EQU    F8+80                               *
000D66 000002D0    F10    EQU    F9+80                               *
000D66 00000320    FKLEN  EQU    F10+80        ARRAY-LENGTH          *
000D66 00000320    CHART  EQU    FKLEN         CHARACTER-TABLE       *
000D66             *-----------------------------------------------*
000D66             *......IMPLEMENTATION-DATA TERMINAL EMULATION....*
000D66             *                                                 *
000D66 00001400    TEMUL  EQU    $1400          FREE FOR TERM.-EMULA*
000D66             *      TEMUL MUST END WITH 00 ! ..................*
000D66             *-----------------------------------------------*
000D66             *..... STAT1-WORD**STATUS-BIT'S ..................*
000D66 00000000    CRTLF  EQU    0              CONTROL FLAG         *
000D66 00000001    SHIFTF EQU    1              SHIFT FLAG           *
000D66 00000002    CAPSF  EQU    2              CAPSLOCK FLAG        *
000D66 00000003    KEYF   EQU    3              KEY FLAG             *
000D66 00000004    ALTF   EQU    4              ALTERNATE FLAG       *
000D66 00000005    SCROLF EQU    5              SMOOTH SCROL FLAG    *
000D66 00000006    CURSF  EQU    6              CURSOR ON FLAG       *
000D66 00000007    ACTION EQU    7              CHAR IN SERVICE      *
000D66             *..... STAT2-WORD**STATUS-BIT'S ..................*
000D66 00000000    AUTOY  EQU    0              DELAY END            *
000D66 00000001    AUTOR  EQU    1              REPEAT END           *
000D66 00000002    CURB   EQU    2              CURSOR BLINK         *
000D66 00000003    HILOW  EQU    3              HIGH/LOW RESOLUTION  *
000D66 00000004    GER    EQU    4              GERMAN-TYP           *
000D66 00000005    ESC    EQU    5              ESC WAS PRESSED      *
000D66 00000006    ESC1   EQU    6              1ST CURSOR-POS.      *
000D66 00000007    ESC2   EQU    7              2ND CURSOR-POS.      *
000D66             *..... STAT3-WORD**STATUS-BIT'S ..................*
000D66 00000000    ESCG1  EQU    0              1ST VIDEO ATTRIB     *
000D66 00000001    REV    EQU    1              CHAR REVERSE         *
000D66 00000002    UND    EQU    2              CHAR UNDERLINE       *
000D66             *-----------------------------------------------*
000D66 0000001E    REP    EQU    30             DELAY RATE           *
000D66 00000003    RATE   EQU    3              REPEAT RATE          *
000D66             *                                                 *
000D66             *-----------------------------------------------*
000D66             *                                                 *
000D66             *......ONE CHARACTER FROM TASTATUR................*
000D66             *                                                 *
000D66             *      A0 : POINTER                               *
000D66             *      A1 : POINTER                               *
000D66             *      A3 : BUFFER-POINTER                        *
000D66             *      D0 : DATA BYTE                             *
```

```
000D66                  *       D1 : COUNTER                                    *
000D66                  *                                                       *
000D66 080603030303
       07030202         MAUST   DC.B    8,6,3,3,3,3,7,3,2,2                     *
000D70                  *                                                       *
000D70 2678083E         KEYINA  MOVEA.L IDP4,A3         BUFFER-INDEX            *
000D74 103900FFFC02             MOVE.B  ACIA1D,D0       RECIEVE TASTATUR KEY*
000D7A 4A2B0016                 TST.B   KEYADR(A3)      BYTECOUNTER == 00 ?     *
000D7E 675E                     BEQ.S   KEYIN4          B:NO STATUS MESSAGE     *
000D80 6014                     BRA.S   KEYIN2                                  *
000D82 1200             KEYIN1  MOVE.B  D0,D1           SAVE                    *
000D84 903C00F6                 SUB.B   =$F6,D0         SET BYTECOUNTER         *
000D88 103B00DC                 MOVE.B  MAUST(D0.W),D0  NUMBER OF BYTES         *
000D8C 17400016                 MOVE.B  D0,KEYADR(A3)   STORE NUMBER OF BYTE*
000D90 422B0017                 CLR.B   KEYADR+1(A3)    CLEAR FOR COUNT         *
000D94 1001                     MOVE.B  D1,D0           RELOAD                  *
000D96 43EB0016         KEYIN2  LEA      KEYADR(A3),A1  ADR FOR KEYBORD DATA*
000D9A 522B0017                 ADDQ.B  =1,KEYADR+1(A3) COUNTER+1               *
000D9E 4241                     CLR     D1                                      *
000DA0 122B0017                 MOVE.B  KEYADR+1(A3),D1 COUNTER                 *
000DA4 B211                     CMP.B   (A1),D1         MORE BYTES ?            *
000DA6 6708                     BEQ.S   KEYIN3          B:IF  LAST BYTE         *
000DA8 13B01001                 MOVE.B  D0,1(A1,D1.W)   STORE BYTE              *
000DAC 6000011E                 BRA     OUTNUL          EXIT                    *
000DB0 13B01001         KEYIN3  MOVE.B  D0,1(A1,D1.W)   STORE LAST BYTE         *
000DB4 122B0016                 MOVE.B  KEYADR(A3),D1   NUMBER OF BYTES         *
000DB8 5301                     SUBQ.B  =1,D1                                   *
000DBA 5489                     ADDQ.L  =2,A1           POINTER TO BYTES        *
000DBC 41F808E6                 LEA     MAUS,A0         POINTER TO MAUS         *
000DC0 10D9             KEYINS  MOVE.B  (A1)+,(A0)+     STORE BYTES             *
000DC2 51C9FFFC                 DBF     D1,KEYINS       B: NOT READY            *
000DC6 422B0016                 CLR.B   KEYADR(A3)      CLEAR FOR NEXT HEADE*
000DCA 4CDF1F03                 MOVEM.L (A7)+,D0/D1/A0/A1/A2/A3/A4 RELOAD *
000DCE 48E74340                 MOVEM.L D1/D6/D7/A1,-(A7) SAVE                  *
000DD2 223C80000000             MOVE.L  =$80000000,D1   SET EVENT CODE          *
000DD8 2278080E                 MOVEA.L PIRTRI,A1       SOFTWARE INTR           *
000DDC 4ED1                     JMP     (A1)            AND EXIT                *
000DDE                  *                                                       *
000DDE 0C0000F6         KEYIN4  CMPI.B  =$F6,D0         STATUS HEADER ?         *
000DE2 629E                     BHI.S   KEYIN1          B: YES                  *
000DE4 22780836                 MOVEA.L IDP2,A1         GET TABEL INDEX         *
000DE8 43E90020                 LEA     TAB(A1),A1      START ADR 1.TABLE       *
000DEC 08800007                 BCLR    =$7,D0          KEY DOWN ?              *
000DF0 66000226                 BNE     UP              B: NO                   *
000DF4                  *-----------------------------------------------------*
000DF4 0C000073         DOWN    CMPI.B  =$73,D0         TO GREAT ?              *
000DF8 6C0000D2                 BGE     OUTNUL          EXIT                    *
000DFC 0C00003A                 CMPI.B  =$3A,D0         KEY CAPSLOCK  ?         *
000E00 660A                     BNE.S   CRTL            B: NO                   *
000E02 086B00020002             BCHG    =CAPSF,STAT1(A3) TOGGLE CAPSFLAG        *
000E08 600000C2                 BRA     OUTNUL          EXIT                    *
000E0C                  *                                                       *
000E0C 082B00000002 CRTL        BTST    =CRTLF,STAT1(A3) CONTROL FLAG TRUE *
000E12 670A                     BEQ.S   ALT             B: NO                   *
000E14 06400180                 ADDI.W  =$180,D0        INDEX TABLE4            *
000E18 10310000                 MOVE.B  $0(A1,D0.W),D0  READ TABLE4             *
000E1C 6060                     BRA.S   OUT             EXIT                    *
000E1E                  *                                                       *
000E1E 082B00040002 ALT         BTST    =ALTF,STAT1(A3) ALTFLAG TRUE ?          *
000E24 6730                     BEQ.S   SHIFT           B: NO                   *
000E26                  *               $1A <= D0 <= 28 ( UE, OE, AE )  ?       *
```

```
000E26                 *                OR D0 EQ 62                          *
000E26 0C00001A                CMPI.B =$1A,D0          IS D0 LT $1A ?       *
000E2A 6D2A                    BLT.S  SHIFT            B: NO                *
000E2C 0C000028                CMPI.B =$28,D0          IS D0 GT $28 ?       *
000E30 6E18                    BGT.S  ALT2             B: NO                *
000E32 0400001A                SUBI.B =$1A,D0             YES               *
000E36 082B00010002 ALT0       BTST   =SHIFTF,STAT1(A3) SHIFTFLAG TRUE ?    *
000E3C 6702                    BEQ.S  ALT1             B: NO                *
000E3E 5440                    ADDQ   =2,D0            INDEX + 2            *
000E40 06400200     ALT1       ADDI.W =$200,D0         INDEX TABLE 5        *
000E44 10310000                MOVE.B $0(A1,D0.W),D0 READ TABLE 5           *
000E48 6034                    BRA.S  OUT              EXIT                 *
000E4A 0C000062     ALT2       CMPI.B =$62,D0          IS HELP HIT ?        *
000E4E 6606                    BNE.S  SHIFT            B: NO                *
000E50 103C0001                MOVE.B =1,D0            MAKE INDEX FOR TABLE*
000E54 60E0                    BRA.S  ALT0                                  *
000E56                 *                                                    *
000E56 082B00010002 SHIFT      BTST   =SHIFTF,STAT1(A3) SHIFT FLAG TRUE     *
000E5C 670A                    BEQ.S  CAPS             B: NO                *
000E5E 06400080                ADDI.W =$80,D0          INDEX TABLE 2        *
000E62 10310000                MOVE.B $0(A1,D0.W),D0 READ TABLE2            *
000E66 6016                    BRA.S  OUT              EXIT                 *
000E68                 *                                                    *
000E68 082B00020002 CAPS       BTST   =CAPSF,STAT1(A3) CAPSLOCK FLAG TRUE*
000E6E 670A                    BEQ.S  SMAL             B: NO                *
000E70 06400100                ADDI.W =$100,D0         INDEX TABLE3         *
000E74 10310000                MOVE.B $0(A1,D0.W),D0 READ TABLE3            *
000E78 6004                    BRA.S  OUT              EXIT                 *
000E7A                 *                                                    *
000E7A 10310000     SMAL       MOVE.B $0(A1,D0.W),D0 READ TABLE1            *
000E7E                 *                                                    *
000E7E                 *......NOW KEY-STRING IS IN D0...................*
000E7E                 *                                                    *
000E7E 6A000190     OUT        BPL    OUTRTS           B:IF ASCII STRING    *
000E82                 *                                                    *
000E82                 *..... NO ASCII OUTPUT.............................*
000E82 37400026                MOVE   D0,RET(A3)       SAVE CHAR            *
000E86 0240001F                ANDI   =$001F,D0        GET TABLE INDEX      *
000E8A E340                    ASL    =1,D0                                 *
000E8C 30380006                MOVE   OUTTAB(D0),D0    LOAD FROM TABLE      *
000E90 4EFB0002                JMP    OUTTAB(D0.W)     JUMP BY TABLE        *
000E94                 *                                                    *
000E94 0038         OUTTAB     DC     OUTNUL-OUTTAB    NOTHING          80 *
000E96 004C                    DC     KEYF1-OUTTAB     FUNCTIONKEY 1    81 *
000E98 0054                    DC     KEYF2-OUTTAB     FUNCTIONKEY 2    82 *
000E9A 005C                    DC     KEYF3-OUTTAB     FUNCTIONKEY 3    83 *
000E9C 0064                    DC     KEYF4-OUTTAB     FUNCTIONKEY 4    84 *
000E9E 006C                    DC     KEYF5-OUTTAB     FUNCTIONKEY 5    85 *
000EA0 0074                    DC     KEYF6-OUTTAB     FUNCTIONKEY 6    86 *
000EA2 007C                    DC     KEYF7-OUTTAB     FUNCTIONKEY 7    87 *
000EA4 0084                    DC     KEYF8-OUTTAB     FUNCTIONKEY 8    88 *
000EA6 008C                    DC     KEYF9-OUTTAB     FUNCTIONKEY 9    89 *
000EA8 0094                    DC     KEYF10-OUTTAB    FUNCTIONKEY 10   8A *
000EAA 00DA                    DC     SHIFS-OUTTAB     KEY SHIFT        8B *
000EAC 0038                    DC     OUTNUL-OUTTAB    CAPSLOCK         8C *
000EAE 00D0                    DC     CTRLS-OUTTAB     CONTROL          8D *
000EB0 0138                    DC     HOME-OUTTAB      HOME             8E *
000EB2 013C                    DC     CHAINS-OUTTAB    INSERT           8F *
000EB4 0108                    DC     BREAK-OUTTAB     UNDO             90 *
000EB6 0124                    DC     HELPTS-OUTTAB    HELP             91 *
000EB8 0148                    DC     LININS-OUTTAB    SHIFT+INSERT     92 *
```

```
000EBA 00E4                  DC      ALTS-OUTTAB     ALTERNATE          93 *
000EBC 00EE                  DC      SCROLS-OUTTAB   CTRL+CURSOR LEFT   94 *
000EBE 00F8                  DC      SMOOTS-OUTTAB   CTRL+CURSOR RIGHT95 *
000EC0 0102                  DC      ABORT-OUTTAB    SHIFT+UNDO         96 *
000EC2 010E                  DC      CHGSIZ-OUTTAB   CTRL+CURDOWN       97 *
000EC4 012E                  DC      HELPSH-OUTTAB   SHIFT+HELP         98 *
000EC6 0154                  DC      CHADEL-OUTTAB   DELETE             99 *
000EC8 0160                  DC      LINDEL-OUTTAB   SHIFT+DELETE       9A *
000ECA 0178                  DC      HCPY-OUTTAB     ALTERNATE+HELP     9B *
000ECC                *                                                    *
000ECC                *.......................................................*
000ECC                *                                                    *
000ECC 4240           OUTNUL CLR     D0              NO ASCII STRING      *
000ECE 37400010              MOVE    D0,CHAR(A3)     CLEAR CHAR           *
000ED2 377C001E0012          MOVE    =REP,AUTOD(A3) SET AUTOREP COUNT     *
000ED8 08AB00000004          BCLR    =AUTOY,STAT2(A3) DISABLE AUTOREP     *
000EDE 4ED4                  JMP     (A4)            EXIT                 *
000EE0                *                                                    *
000EE0 247C00000000   KEYF1  MOVEA.L =F1,A2          LOAD POINTER         *
000EE6 6046                  BRA.S   KEYFA           GO ON                *
000EE8 247C00000050   KEYF2  MOVEA.L =F2,A2          LOAD POINTER         *
000EEE 603E                  BRA.S   KEYFA           GO ON                *
000EF0 247C000000A0   KEYF3  MOVEA.L =F3,A2          LOAD POINTER         *
000EF6 6036                  BRA.S   KEYFA           GO ON                *
000EF8 247C000000F0   KEYF4  MOVEA.L =F4,A2          LOAD POINTER         *
000EFE 602E                  BRA.S   KEYFA           GO ON                *
000F00 247C00000140   KEYF5  MOVEA.L =F5,A2          LOAD POINTER         *
000F06 6026                  BRA.S   KEYFA           GO ON                *
000F08 247C00000190   KEYF6  MOVEA.L =F6,A2          LOAD POINTER         *
000F0E 601E                  BRA.S   KEYFA           GO ON                *
000F10 247C000001E0   KEYF7  MOVEA.L =F7,A2          LOAD POINTER         *
000F16 6016                  BRA.S   KEYFA           GO ON                *
000F18 247C00000230   KEYF8  MOVEA.L =F8,A2          LOAD POINTER         *
000F1E 600E                  BRA.S   KEYFA           GO ON                *
000F20 247C00000280   KEYF9  MOVEA.L =F9,A2          LOAD POINTER         *
000F26 6006                  BRA.S   KEYFA           GO ON                *
000F28 247C000002D0   KEYF10 MOVEA.L =F10,A2         LOAD POINTER         *
000F2E D5F8081E       KEYFA  ADDA.L FUNKP,A2         POINTER READY        *
000F32 48E71004              MOVEM.L D3/A5,-(A7)     SAVE                 *
000F36 2A4A                  MOVEA.L A2,A5           PTR                  *
000F38 4243           KEYFIN CLR     D3                                    *
000F3A 161D                  MOVE.B (A5)+,D3         DATA LENGTH          *
000F3C 08EB00030002          BSET    =KEYF,STAT1(A3) SET KEYFLAG          *
000F42 51CB0016       KEYFB  DBF     D3,KEYFC        B: IF DATA NOT END   *
000F46 1015                  MOVE.B (A5),D0          LAST DATA            *
000F48 49FAF9F6              LEA     IR6Z01,A4       EXIT ADRESS          *
000F4C 4CDF2008              MOVEM.L (A7)+,D3/A5     RELOAD               *
000F50 376B00260010          MOVE    RET(A3),CHAR(A3) FOR AUTOREPEAT      *
000F56 6000F764              BRA     IR6I02          IR EXIT              *
000F5A 101D           KEYFC  MOVE.B (A5)+,D0         READ DATA            *
000F5C 49FAFFE4              LEA KEYFB,A4            RETURN ADRESS        *
000F60 6000F75A              BRA     IR6I02          SEND DATA TO CMMD    *
000F64                *                                                    *
000F64 08EB00000002   CTRLS  BSET    =CRTLF,STAT1(A3) SET CONTROL FLAG    *
000F6A 6000FF60              BRA     OUTNUL          EXIT                 *
000F6E 08EB00010002   SHIFS  BSET    =SHIFTF,STAT1(A3) SET SHIFT FLAG     *
000F74 6000FF56              BRA     OUTNUL          EXIT                 *
000F78 08EB00040002   ALTS   BSET    =ALTF,STAT1(A3) SET.ALTERNATE FLAG   *
000F7E 6000FF4C              BRA     OUTNUL                               *
000F82 08AB00050002   SCROLS BCLR    =SCROLF,STAT1(A3) SET SCROL          *
000F88 6000FF42              BRA     OUTNUL                               *
```

```
000F8C 08EB00050002 SMOOTS BSET    =SCROLF,STAT1(A3) SET SMOOTH SCROL *
000F92 6000FF38            BRA     OUTNUL                             *
000F96 2278007C     ABORT  MOVEA.L $7C,A1          FETCH ABORT ADR    *
000F9A 4ED1                JMP     (A1)                               *
000F9C 4240         BREAK  CLR     D0              BREAK CONDITION    *
000F9E 4EFAF826            JMP     IR6U00                             *
000FA2 30280008     CHGSIZ MOVE    COLOUR(A3),D0                      *
000FA6 5240                ADDQ    =1,D0           PEN-COLOUR + 1     *
000FA8 02400003            ANDI    =$0003,D0       MAX PEN EQ 3       *
000FAC 6602                BNE.S   CHGSI1          B: NOT ZERO        *
000FAE 5240                ADDQ    =1,D0                              *
000FB0 37400008     CHGS11 MOVE    D0,COLOUR(A3)   SAVE               *
000FB4 6000FF16            BRA     OUTNUL                             *
000FB8 21F8081A084A HELPTS MOVE.L VIDEOP,IDP7   VIDEOP TO SCREEN 0    *
000FBE 6000FF0C            BRA     OUTNUL                             *
000FC2 21F8083A084A HELPSH MOVE.L IDP3,IDP7    VIDEOP TO LAST SCREEN *
000FC8 6000FF02            BRA     OUTNUL                             *
000FCC 6000FEFE     HOME   BRA     OUTNUL          NO ACTION YET      *
000FD0 48E71004     CHAINS MOVEM.L D3/A5,-(A7)     SAVE               *
000FD4 4BFA002A            LEA     CHAINT,A5       LOAD POINTER       *
000FD8 6000FF5E            BRA     KEYFIN          MAKE THE INPUT     *
000FDC 48E71004     LININS MOVEM.L D3/A5,-(A7)     SAVE               *
000FE0 4BFA0021            LEA     LININT,A5       LOAD POINTER       *
000FE4 6000FF52            BRA     KEYFIN          MAKE THE INPUT     *
000FE8 48E71004     CHADEL MOVEM.L D3/A5,-(A7)     SAVE               *
000FEC 4BFA0018            LEA     CHADET,A5       LOAD POINTER       *
000FF0 6000FF46            BRA     KEYFIN          MAKE THE INPUT     *
000FF4 48E71004     LINDEL MOVEM.L D3/A5,-(A7)     SAVE               *
000FF8 4BFA000F            LEA     LINDET,A5       LOAD POINTER       *
000FFC 6000FF3A            BRA     KEYFIN          MAKE THE INPUT     *
001000 01180C       CHAINT DC.B    $01,$1B,$0C     <ESC FF>           *
001003 01180A       LININT DC.B    $01,$1B,$0A     <ESC LF>           *
001006 011808       CHADET DC.B    $01,$1B,$08     <ESC BS>           *
001009 01180B       LINDET DC.B    $01,$1B,$0B     <ESC VT>           *
00100C              *-----------------------------------------------*
00100C 6000FE8E     HCPY   BRA     OUTNUL                             *
001010              *-----------------------------------------------*
001010              *                                                 *
001010 37400010     OUTRTS MOVE    D0,CHAR(A3)     SAVE CHAR AUTOREP  *
001014 6000F6A6            BRA     IR6102                             *
001018              *                                                 *
001018              *-----------------------------------------------*
001018              *                                                 *
001018 10310000     UP     MOVE.B  $0(A1,D0.W),D0                     *
00101C 0C000080            CMPI.B  =$80,D0         KEY CONTROL        *
001020 660A                BNE.S   SHIFTU                             *
001022 08AB00000002        BCLR    =CRTLF,STAT1(A3) RESET CONTROL FLAG*
00102B 6000FEA2            BRA     OUTNUL                             *
00102C              *                                                 *
00102C 0C000088     SHIFTU CMPI.B  =$88,D0         KEY SHIFT          *
001030 660A                BNE.S   ALTU                               *
001032 08AB00010002        BCLR    =SHIFTF,STAT1(A3) RESET SHIFT FLAG *
001038 6000FE92            BRA     OUTNUL                             *
00103C              *                                                 *
00103C 0C000093     ALTU   CMPI.B  =$93,D0         KEY ALTERNATE      *
001040 6600FE8A            BNE     OUTNUL                             *
001044 08AB00040002        BCLR    =ALTF,STAT1(A3) RESET ALTERNATE FLA*
00104A 6000FE80            BRA     OUTNUL                             *
00104E              *                                                 *
00104E              ***************************************************
00104E              * 80            NOTHING                           *
```

```
00104E                * 81 - 8A     FUNCTIONKEYS                          *
00104E                * 8B          SHIFT                                 *
00104E                * 8C          CAPSLOCK                              *
00104E                * 8D          CONTROL                               *
00104E                * 8E          HOME                                  *
00104E                * 8F          INSERT                                *
00104E                * 90          UNDO = BREAK                          *
00104E                * 91          HELP = VIDEOP TO SCREEN 0             *
00104E                * 92          LOCAL                                 *
00104E                * 93          ALTERNATE                             *
00104E                * 94          CTRL + CURSOR LEFT       SCROL        *
00104E                * 95          CTRL + CURSOR RIGHT      SMOOTH SCRO*
00104E                * 96          SHIFT + UNDO  = ABORT                 *
00104E                * 97          CTRL + CURDOWN = CHANGE CHAR SIZE     *
00104E                * 98          SHIFT + HELP =VIDEOP TO LAST SCREEN*
00104E                *                                                   *
00104E                ***********************************************
00104E R0000000             END        End of Implementation Slice      *

0 ERRORS
```

Schlagwortindex

A(ctivate) B-II-1, E-I-3f., E-I-55f.
A-Format C-III-15
A1:, A2: etc. A-II-1f.
Abbruch einer Task E-I-50f.
Abbruch-Anweisung C-III-10
Ablauf-Steueranweisung C-III-9
ACOS C-III-17
AFTER B-II-2
AI (-Parameter) C-I-5, E-III-2
Aktivieren B-II-1, E-I-3f., E-I-51f.
ALL B-II-3
Anhalte-Anweisung C-III-10
Anweisung (PEARL) C-III-8
Anweisung (RTOS) B-I-1f.
Anweisungsformat B-I-1
APPEND C-III-20
Arcus C-III-17
Argumentliste C-III-9
Array C-I-6
AS(SEM) B-II-4
ASIN C-III-17
Assembler B-II-4, D-I-1f.
Assembler-Unterprogramme C-IV-1, D-III-1, E-IV-1
ASSIGN C-III-24
AT B-II-6
ATAN C-III-17
Ausdruck (Assembler) D-I-4
Ausdruck (PEARL) C-III-8
Ausgabe-Formatierung C-III-14f.
Ausgabeumlenkung B-II-40, C-III-24
Ausplanung C-III-10, E-I-37
Autolink E-VII-1

B-Format (Bitketten) C-III-15
B1:, B2: etc. A-II-1f.
Basic (Datenstationstyp) A-II-5
Baudrate F-I-6
Bedienbefehl-Definition E-VII-13f.
Bedieninterpreter A-I-1, B-I-1f.
Bedientask B-I-1f.
Betreuungstask A-I-9, A-II-9, E-I-60, E-III-1f.
Bezeichner (PEARL) C-I-1
Big-Module-Option C-II-3
Binär-Transfer A-II-8, C-III-21
Binär-Ziffer C-III-6
Bitketten-Konstante C-III-8
Block C-III-8
Break A-I-1
Break Point >Einzelschritt
BU A-II-5
Buchstabe C-III-6

C(ontinue) B-II-11
C1:, C2: etc. A-II-1f.

CD B-II-7
CF B-II-8
Character Scan E-I-10
CLOCK B-II-9, E-I-41, E-I-49
Codeprotokollierung C-II-1
Communication Element E-I-19, E-I-42, E-I-65, E-III-1f.
Compiler B-II-41, C-I-1f.
Compiler-Mnemonics E-IV-10f.
Continue (Traps) E-I-7f.
COPY A-I-5, B-II-12
Cosinus C-III-17
Cursorpositionierung F-I-2, F-I-14

D-Format C-III-16
DATE B-II-15
Datentypen C-I-1f., C-III-6
DATION, Datenstation A-I-9, A-II-1f., B-II-53, C-I-4, C-III-7, E-VII-11
Datum B-II-15f.
DD B-II-17
Definition von Datenstationen C-I-4
Delimiter-Test E-I-12
Device-Parameter B-II-53, E-I-14, E-VII-12
Dimension C-III-7
DIN-Abweichungen C-I-3
Directory B-II-7, B-II-18
Direktiven (Assembler) D-I-3
Disable Interrupt B-II-19, E-I-13
Disassembler F-II-7
Diskettenbetrieb A-II-9
Dispatcher A-I-1, E-I-35,
DL B-II-20
DM B-II-21
DR B-II-22
DRANF C-III-25
Drive >Untergliederungsnummer
Druckerschnittstelle A-II-11
Duplexbetrieb A-II-1f.
Dyadische Operatoren C-III-11

E-Format C-III-14
E/A-Objekt C-III-9
Echtzeit-Anweisung >Einplanung, C-III-9
ED B-II-23
Editor (ED:) A-I-2, A-II-7, B-II-23
Ein-/Ausgabe-Formatierung C-III-14f.
Ein-/Ausgabeanweisung C-III-9
Ein-/Ausgabetreiber E-V-1, E-VI-1f.
Ein-/Ausgabeumlenkung C-III-24
Einbaufunktion C-III-13, C-III-17f.
Eingabe-Formatierung C-III-14f.
Eingabefeld C-I-5
Einplanung B-II-1f., C-III-9f., E-I-51f.
Einplanungen für Bedienbefehle B-III-1
Einzelschritt-Modus B-II-60, E-I-32f., F-II-19, F-II-22
Enable Interrupt B-II-27, E-I-15
Entfernen von Modulen/Tasks B-II-60 >UNLOAD
Ereignis-Operation >Interrupt

Error Message (Trap) E-I-15
Error-Task A-IV-1
Exponential-Funktion C-III-17
Extern-Symbol E-VII-20

F-Format C-III-14
F0:, F1: etc. A-II-9
Fehlermeldungen (Assembler) D-II-1
Fehlermeldungen (Compiler) C-V-1
Fehlermeldungen (Laufzeit) C-VI-1
Fehlermeldungen (Monitor) F-II-25
Fehlermeldungen (RTOS) A-IV-1
Feldindextest C-II-2
Feldzugriff C-I-6
FILES B-II-26
Filesystem E-II-1f.
Floppy A-II-8, E-II-1
FORM B-II-29
Formate (Floppy, Winchester) B-II-29, E-II-1
Formatieren (Diskette) A-I-5, B-II-29
Formatliste C-III-8
Fortran-Unterprogramme C-IV-1
Fortsetz-Anweisung C-III-10
FREE B-II-28, B-II-30
Funktion C-III-7, C-III-9
Funktionstasten (Atari ST) F-I-4

Genauigkeit C-III-6
Gerätebenennung C-III-6
GETPIX C-III-19
GO B-II-31, F-II-12
Grafik (Basisfunktionen) C-III-19
Grafik (Einbaufunktionen) F-I-8f.

H0:, H1: etc. A-II-9
Hardcopy (Atari ST) F-I-5
Harddisk >Winchester
Header-Text E-VII-19
Hexadezimal-Ziffer C-III-6
Hyperprozessor D-I-2, E-IV-7f.

I/O-Operationen C-III-20, E-III-1f.
Identitätsfestlegung C-III-8
IDF (OPEN ... BY -) C-I-5, C-III-24
Idle A-I-1
Indexliste C-III-8
Indextest-Option C-II-2
Indextest-Traps E-I-26f.
Initialisieren von Variablen C-I-1
Interrupt A-II-14, B-II-19, B-II-27, B-II-52, B-II-64,
C-III-9, E-I-13f., F-I-7
Interrupt-Kodierung A-II-14, E-VI-3
Interruptpuffer E-VII-10
Interruptvektoren E-VII-17

Komponente C-III-7
Komprimierung A-II-7
Konstante C-III-7

Konvertierung E-I-6
Koordinations-Anweisung C-III-10

L(ist) A-I-1, B-II-32
L0:, L1: etc. A-II-10
Label-Feld (Assembler) D-I-2
Laden/Linken >LOAD
Laufzustand >Taskstatus
LD (Datenstation) C-I-4
LD (Log. dualis) C-III-17
LDN >Warteschlange
Leerzeichen E-I-46
Length C-I-2, C-III-6
LG C-III-17
Limit-Error C-I-1
LINE C-III-19
Linken >LOAD
List-Format C-III-16
LN C-III-17
LOAD B-II-34
Logarithmus C-III-17
LU B-II-37
Längenbegrenzung für Module C-I-1, >Big-Module-Option
Längenfestlegung C-III-6
Löschen einer Datei B-II-50
Löschen einer Einplanung B-II-43

Markierungsoption C-II-2
Mathematische Funktionen C-III-17
Matrix C-III-7
Mehrnutzer-Betrieb >Multiuser-Betrieb
Merkmal (Geräte-) C-III-6
Mischen von Dateien B-II-13
MKDIR B-II-38
Mode-Wort (CE) E-III-2
Modifikation von RTOS-UH E-VII-1f.
Modul C-III-6
Modul-Identifizierung E-I-21
modulglobal C-III-7
Modulgröße C-I-1
Modulkopf C-III-6, D-III-1
Modulvariable C-III-7, E-VII-16
Monadische Operatoren C-III-11
Monitor F-II-1f.
Multiuser-Betrieb A-I-7

Namenliste C-III-6
NE (No error-message flag) C-I-4
NOTRACE B-II-39
NOW C-III-23
Nukleus E-VII-1

O (Ausgabeumlenkung) B-II-40
Oktal-Ziffer C-III-6
Operanden-Feld (Assembler) D-I-4
Operations-Feld (Assembler) D-I-2
Operatoren (Assembler) D-I-5
Operatoren (PEARL) C-III-4, C-III-11

P(EARL) B-II-41
P-Bus A-II-5
PAGE C-III-16
Parallele Schnittstelle A-II-11
Parameter (Prozedur-) C-III-8
Parameterbeschreibung C-III-7
Parameterübergabe (Assembler) E-IV-4f.
Parametrierung von Datenstationen B-II-53, E-VII-12
PEARL-Programm C-III-6
PEARL-Schlüsselwörter C-III-1
Peripherie >Datenstationen
PP: A-II-11
Prevent B-II-43
PREVENT B-II-43
Printer-Port A-I-10, A-II-11
Priorität A-I-2, C-III-10
Problemteil C-III-6
Produktionsregeln C-III-5f.
PROM B-II-44f., C-I-2
Prozedur C-III-7, E-I-36, E-I-45
Prozedur-Kopf C-III-8
Prozeßinterrupt >Interrupt
Prozeßumschalter A-I-1
PWD B-II-46

Quadratwurzel C-III-17

R-Format C-III-16
RANF C-III-25
READ C-III-21
REF-Variable C-I-7
REFADD C-III-23
RELCE E-III-1f.
RELEASE B-II-47
Reset A-I-1
RESUME E-I-59
RETURN B-II-48
REWIND B-II-49, C-III-20
RM B-II-50
RMDIR B-II-51
ROM-Code B-II-44, C-I-2
RTOS-Kommandos B-I-4, B-II-1f.
Rücksprung-Anweisung C-III-9

S(torage) B-II-52
S-Records D-I-6
SAVEP C-III-20
Scan E-I-40
Scheibe E-I-48, E-VII-1f.
Schließen von Dateien A-I-6, B-II-48
Schlüsselwörter (PEARL) C-III-1
Schnittstellen A-I-9, A-II-1f.
SD B-II-53
Sedezimal >Hexadezimal
SEEK C-III-20
Semaphore B-II-47, E-I-43f.
SEND A-II-5f.

Sequentielle Ausführung A-I-6, B-I-3, B-III-1
Serielle Schnittstellen A-II-1f.
SETPIX C-III-19
SHARE B-II-54
Shell >Bedieninterpreter
SHOW B-II-55
Signale C-I-3
Signalmarke E-VII-1f.
Sinus C-III-17
Size-Statement C-I-1
SKIP C-III-16
SM B-II-56
Sonderzeichen C-III-6
Speicherblöcke A-III-1
Speicherverwaltung A-III-1, B-II-49, E-I-47, E-I-62, E-VII-10f.
Spezifikation C-III-7
Sprachumfang C-I-1f.
SQRT C-III-17
ST C-III-18
Statusabfrage von Datenstationen C-III-18
Steuerelemente (Format) C-III-16
SU(SPEND) B-II-57
Subdirectory >Working Directory
Subtask-Erzeugung E-I-20
Suchen F-II-13
SYNC B-II-58, C-III-20
Syntax (RTOS-UH/PEARL) C-III-5
Systemadressen (Atari ST) F-I-7, F-III
Systemkonfigurierung E-VII-1f.
Systemtask-Definition E-VII-8
Systemteil C-I-4, C-III-6
Systemuhr B-II-9

T(ERMINATE) B-II-59
T-Format C-III-15
TAKE A-II-5
Tangens C-III-17
Task C-III-10
Task-Identifizierung E-I-24f.
Task-Kommunikation A-II-12
Task-Steueranweisung C-III-9
Taskkopf A-I-2, C-III-10, D-III-1
TASKST C-III-26
Taskstatus A-I-2, B-II-32f., C-III-26
Tastatur (Atari ST) F-I-1
Terminal A-I-1, A-I-3, A-II-1f., B-II-23
Terminal-Emulation (Atari ST) F-I-2
Terminieren >Abbruch-Anw., B-II-59
Texteditor A-II-7, B-II-23f.
Textformatierung C-III-14f.
TFU C-I-4, C-III-22
Time Sharing B-II-54
Trace B-II-60, F-II-23
Traps E-I-1f., E-VII-17
TRIGGER B-II-62, E-I-61
Typdefinition C-III-6

Übersetzungsprotokoll C-II-1
Uhr B-II-9
Umlenkung (E/A) C-III-24
UNLOAD A-I-4, B-II-63
Unterbus A-II-5
Untergliederungsnummer A-I-9f., A-II-9

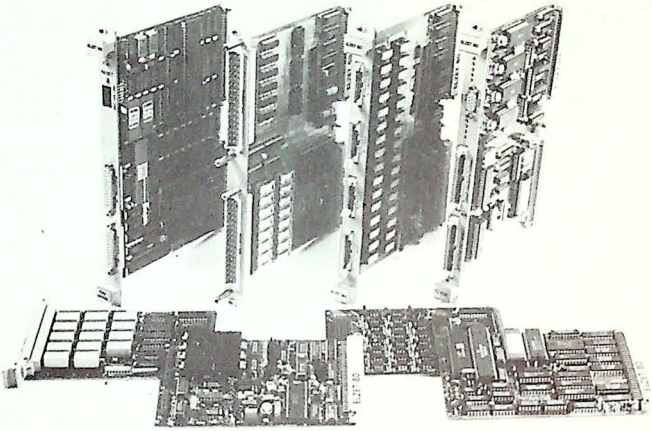
Verbund C-III-6
Verschieben F-II-14
Verzögerte Einplanung B-II-2
VI: (Virtual Input) A-II-12
Virtueller Prozessor E-IV-7f.
VO: (Virtual Output) A-II-12
Voreinstellung C-III-7

Warmstart-Initialisierungscode E-VII-18
Warteschlange A-I-9, A-II-1f., A-II-9, E-I-23, E-I-60, E-III-1f.
WHEN B-II-64
Winchester A-II-12, E-II-1f.
Working Directory A-I-2, B-II-7, B-II-38, B-II-46, B-II-51
Workspace E-I-47, E-I-62f.
WRITE C-III-21

X (Steuerelement) C-III-16
X-on/X-off A-II-1f.
XC: A-I-7, A-II-13, B-I-1f.
XIO E-III-1f.

Zeichen C-III-6
Zeichenketten-Konstante C-III-8
Zeigervariable C-I-7
Zeit-Einplanung C-III-10
Zeitdauer-Konstante C-III-7
Ziffer C-III-6
Zufallszahlen C-III-25
Zugriffsausdruck C-III-8
Zuweisung C-III-8
Zyklische Einplanung B-II-3, C-III-10

RTOS-UH/PEARL
auf VMEbus und ECB



Sie kennen jetzt RTOS-UH und PEARL.

Für den industriellen Einsatz von PEARL empfehlen wir Ihnen industrielle Bussysteme.

Auf VMEbus mit der CPU68000 und auf ECB mit 68008.

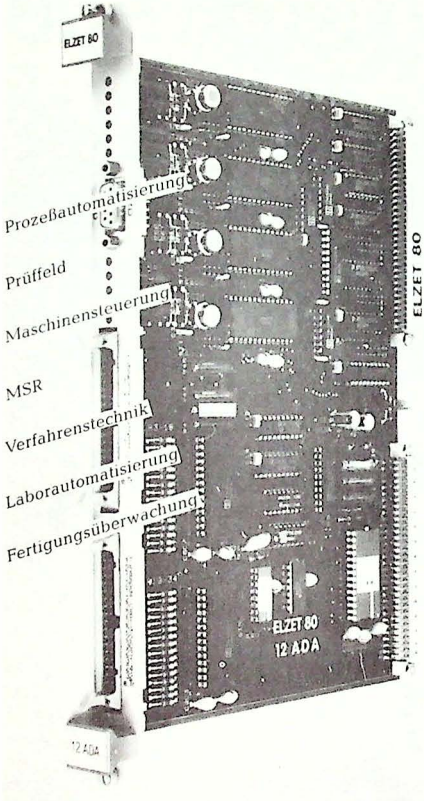
Oder als Einplatinencomputer für ganz kleine Lösungen.

ELZET 80 liefert seit 1978 modulare Mikrocomputersysteme. Weit über 100 Baugruppen für alle Anwendungsbereiche gehören zu unserem Standardprogramm. Für Spezialanwendungen fertigen wir kundenspezifische Lösungen. Vertrauen Sie unserer Erfahrung!

Wir senden Ihnen gerne Kataloge.

ELZET 80
Mikrocomputer GmbH & Co. KG
Wilhelm-Mellies-Straße 88
D-4930 Detmold 18
Tel. 0 52 32 - 81 31 · Tx. 931 473 elzet d

Zum Beispiel: 12 ADA



Zu VMEbus, ECB und RTOS beraten Sie gerne auch unsere Ingenieurbüros (mit Lager):

ESmed GmbH
Fehlerstraße 5
1000 Berlin 41
Tel. 030/8519 00

MEK GmbH
Düppelstraße 71
2300 Kiel 1
Tel. 0431/8042 20

GMS mbH
Marquardstraße 62
6000 Frankfurt 90
Tel. 069/7887 52

Meditec-Data GmbH
Villastraße 9
7000 Stuttgart 1
Tel. 0711/2831 03

PTL GmbH
Schulstraße 28
8000 München 19
Tel. 089/1699 77

Schweiz:
Bernhard-Elektronik
Aarauer Straße 20
CH-5734 Reinach AG
Tel. 064/7169 44

- *12-Bit-A/D-Umsetzer*
- *12 Bit Linearität*
- *16 o. 32 Eingänge*
- *Wahlweise 10/15/25 µs max. Umsetzzeit*
- *Eingangsbereiche: 0..10V, ±5V, ±10V*
- *Int. o. ext. Triggerung*
- *S&H mit 3,5 µs*
- *Vier 12-Bit-D/A's*
- *Spannungsausgänge: 0..5V, 0..10V, ±2,5V, ±5V, ±10V*
- *Max. Lin.fehler 1/2 LSB*

Einzelpreis komplett A/D mit 25 µs und 4 Kanäle D/A:

2277,72 DM
(1998,– DM + 14 % MWSt.)

ELZET 80
Mikrocomputer GmbH & Co. KG
Wilhelm-Mellies-Straße 88
D-4930 Detmold 18
Tel. 05232 - 8131 · Tx. 931473 elzet d